68 MICRO JOURNAL

Australia A $4.75 | New Zealand NZ $ 6.50
Singapore S $9.45 | Hong Kong H $23.50
Malaysia M $9.45 | Sweden 30:-SEK

$2.95 USA

68000

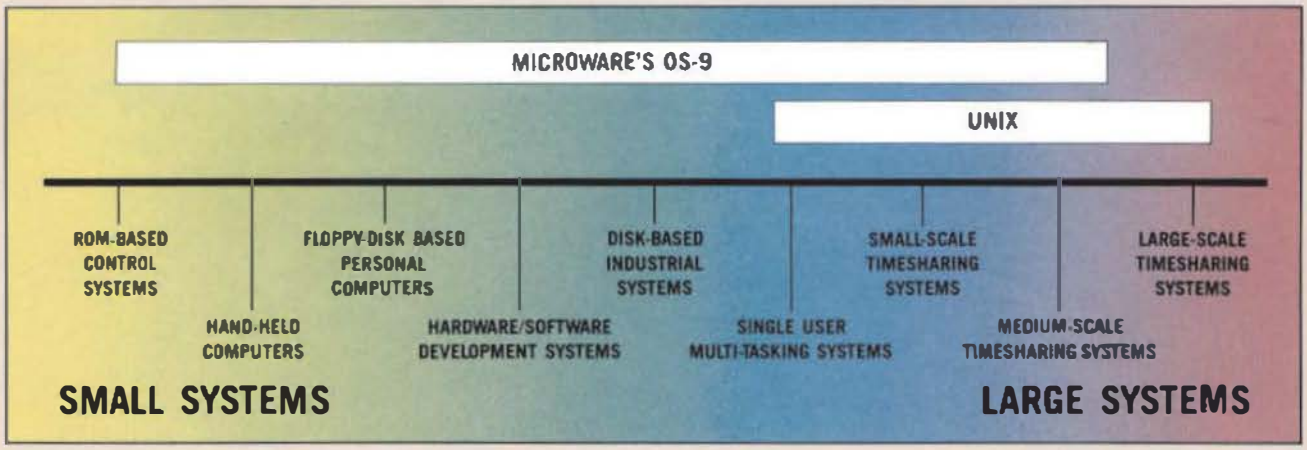
6809

VOLUME VII ISSUE VIII • Devoted to the 68XX User • August 1985
"Small Computers Doing Big Things"

SERVING THE 68XX USER WORLDWIDE

PHOTO © EDIT - NASA

Only Microware's OS-9 Operating System Covers the Entire 68000 Spectrum



Is complicated software and expensive hardware keeping you back from Unix? Look into OS-9, the operating system from Microware that gives 68000 sy tems a Unix-style environment with much less overhead and complexity.

OS-9 is versatile, inexpensive, and delivers outstanding performance on any size system. The OS-9 executive is much smaller and far more efficient than Unix b cause it's written in tast, compact assembly language, making it ideal for critical real-time applications. OS-9 can run on a broad range of 8 to 32 bit systems based on the 68000 or 6809 family MPUs from ROM-based industrial controllers up to large multiuser systems.

OS-9'S OUTSTANDING C COMPILER IS YOUR BRIDGE TO UNIX

Microware's C compiler technology is another OS-9 advantage. The compiler produces extremely fast, compact, and ROMable code. You can easily develop and port system or application software back and forth to standard Unix systems. Cross-compiler versions for VAX and PDP-11 make coordinated Unix/OS-9 software development a pleasure.

SUPPORT FOR MODULAR SOFTWARE — AN OS-9 EXCLUSIVE

Comprehensive support for modular software puts OS-9 a generation ahead of other operating systems. It multiplies programmer productivity and memory efficiency. Application software can be built from individually testable software modules including standard "library" modules. The modular structure lets you customize and reconfigure OS-9 for sp cific hardware easily and quickly.

A SYSTEM WITH A PROVEN TRACK RECORD

Once an underground classic, OS-9 is now a solid hit. Since 1980 OS-9 has been ported to over a hundred 6809 and 68000 systems under license to some of the biggest names in the business. OS-9 has been imbedded in numerous consumer, industrial, and OEM products, and is supported by many independent software suppli rs.

Key OS-9 Features At A Glance

- Compact (16K) ROMable executive written in assembly language
- User "shell" and complete utility set written in C
- C-source code level compatibility with Unix
- Full Multitasking/multiuser capabilities
- Modular design - extremely easy to adapt, modify, or expand
- Unix-type tree structured file system
- Rugged "crash-proof" file structure with record locking
- Works well with floppy disk or ROM-based systems
- Uses hardware or software memory management
- High performance C, Pascal, Basic and Cobol compilers

microware® OS-9

AUTHORIZED MICROWARE DISTRIBUTORS

AUSTRALIA
MICROPROCESSOR CONSULTANTS
16 Bandera Avenue
Wagga Wagga 2650
NSW Australia
phone: 616-931-2331

ENGLAND
VIVAWAY LTD.
36-38 John Street
Luton, Bedfordshire
England LU1 2JE
phone: (0582) 423425
telex: 825115

JAPAN
MICROWARE JAPAN LTD.
3-8-9 Baraki, Ichikawa
Chiba 272-01, Japan
phone: 0473 (26) 4493
telex: 781-299-3122

SWEDEN
MICROMASTER SCANDINAVIAN AB
St Persgatan 7
Box 1309
S751-43 Uppsala
Sweden
phone: 018-138595
telex: 76129

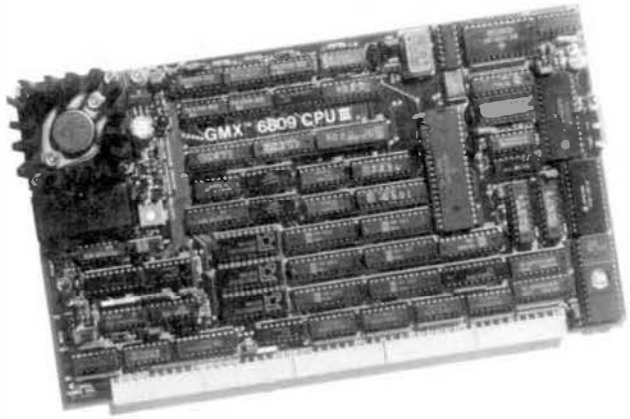
SWITZERLAND
ELSOFT AG
Bankstrasse 9
5432 Neuenhof
Switzerland
phone: (41) 056-862724
telex: 57136

USA
MICROWARE SYSTEMS CORPORATION
1866 NW 114th Street
Des Moines, Iowa 50322
USA
phone: 515-224-1929
telex: 910-520-2535
FAX: 515-224-1352

WEST GERMANY
DR. KEIL GMBH
Porphystrasse 15
D-6905 Schriesheim
West Germany
phone: (0 62 03) 67 41
telex: 465025

OS-9 is a trademark of Microware and Motorola. Unix is a trademark of Bell Labs.

GIMIX STATE OF THE ART 6809 SYSTEMS FOR THE SERIOUS USER.



GIMIX has 19MB or high performance 47MB Winchester Drive Systems and/or Floppy Disk Drive Systems.

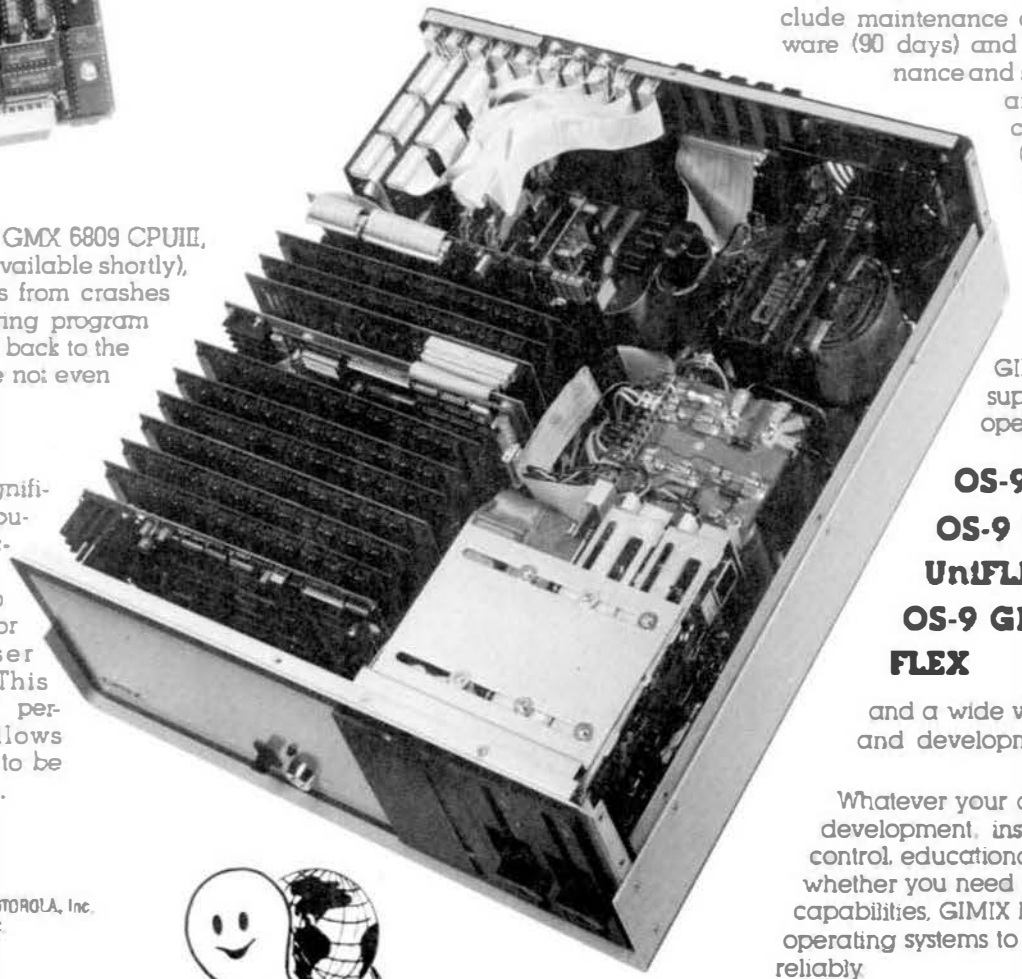
For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer.

GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

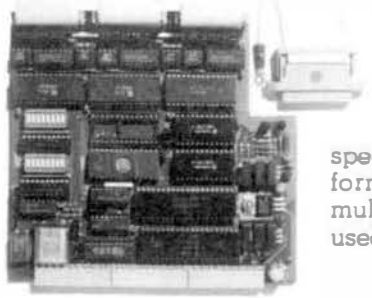
GIMIX versions of OS9 and UniFLEX include maintenance and support by Microware (90 days) and TSC (1 year). Maintenance and support after this period are available at extra cost.

(NOTE: this support and maintenance is only for use with approved GIMIX hardware)

For the ultimate in performance, the Unique GMX 6809 CPUIII, using either OS-9-GMXIII or UniFLEX GMXIII (available shortly), gives protection to the system and other users from crashes caused by defective user programs. e.g. During program development, a programmer who crashes goes back to the shell or the debugger, while the other users are not even aware anything occurred.



The intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions, thereby freeing up the host CPU for running user programs. This speeds up system performance and allows multiple terminals to be used at 19.2K baud.



GIMIX 6809 systems support five predominant operating systems:

OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX

and a wide variety of languages and development software.

Whatever your application: software development, instrumentation, process control, educational, scientific or business; whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably.

Please phone or write if you need further information.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc.
FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.
GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GIMIX INC.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

© 1983 GIMIX Inc.

'68' MICRO JOURNAL

Portions of the text for '68' Micro Journal were prepared using the following furnished Hard/Software:

COMPUTERS - HARDWARE

Southwest Technical Products
219 W. Rhapsody
San Antonio, TX 78216
SO9 - 5/8 DMF Disk - COSI - 8212W - Sprint 3 Printer

GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe - OS9 - FLEX - Assorted Hardware

EDITORS - WORD PROCESSORS

Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX - Editor - Text Processor

Great Plains Computer Co., Inc.
PO Box 916
Idaho Falls, ID 83401
Stylograph - Mail Merge - Spell

Editorial Staff

Don Williams Sr. — Publisher
Larry E. Williams — Executive Editor
Tom E. Williams — Production Editor
Robert L. May — Technical Editor

Administrative Staff

Mary Robertson — Office Manager
Penny Williams — Subscriptions
Christine Kacher — Accounting

Contributing Editors

Ron Anderson — Norm Commo
Peter Dibble — William E. Fisher
Dr. Theo Elbert — Carl Mann
Dr. E. M. Pass — Ron Voigts
Phillip Lucido

Special Technical Projects

Clay Abrams K6AEP
Tom Hunt

CONTENTS

Vol.VIII,Issue VII — August 85

Send All Correspondence To:

Computer Publishing Center
68' Micro Journal
5900 Cassandra Smith Rd.
Hixson, Tn. 37343

Phone (615) 842-4600 or Telex 558 414 PVT BTH

Copyrighted 1985 by Computer Publishing Inc.

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tn. and additional entries. Postmaster: send form 3597 to 68' Micro Journal, POB 849 Hixson, Tn. 37343.

Subscription Rates

1 Year $24.50 U.S.A., Canada & Mexico Add $9.50 a Year. Other Foreign Add $12 a Year for Surface, Airmail Add $48 a Year. Must be in U.S. currency.

Items or Articles For Publication

Articles submitted for publication should include authors name, address, telephone number and date. Articles should be on either 5 or 8 inch disk in STYLOGRAPH or TSC Editor format with 3.5 inch Column width. All disks will be returned. Articles submitted on paper should be 4.5 inches in width (including Source Listings) for proper reductions. Please Use A Dark Ribbon!! No Blue Ink!! Single space on 8X11 bond or better grade paper. No hand written articles accepted. Disks should be in FLEX2 6800 or FLEX9 6809 any version or OS-9 any version.

The following TSC Text Processor commands ONLY should be used: .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. We will enter the rest at time of editing.

All STYLOGRAPH commands are acceptable except .pg page command. We print edited text files in continous text form.

Letters To The Editor

All letters to the editor should comply with the above requirements and must be signed. Letters of "gripes" as well as "praise" are solicited. We reserve the right to reject any submission for lack of "good taste" and we reserve the right to define "good taste".

Advertising Rates

Commercial advertisers please contact 68' Micro Journal advertising department for current rate sheet and requirements.

Classified Advertising

All classified ads must be non-commercial. Minimum of $9.50 for first 20 words and .45 per word after 20. All classifieds must be paid in advance. No classified ads accepted over the phone.

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble
As Published in 68 Micro Journal

The publishers of 68 Micro Journal are proud to announce the publication of Peter Dibbles **OS9 USER NOTES**.

Information for the BEGINNER to the PRO, Regular or CoCo OS9

Using OS9
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

Program interfacing to OS9
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming Languages
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include
No typing all the Source Listings in. Source Code and, where applicable, assembled or compiled **Operating Programs.** The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing **your OWN** more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

BOOK Typeset -- w/ Source Listings (3-Hole Punched; 8 x 11) **$9.95**

Deluxe Binder - - - - - - - - - - $5.50

All Source Listings on Disk
1 8" SS, SD Disk - - - - $14.95
2 5" SS, DD Disks - - - $24.95

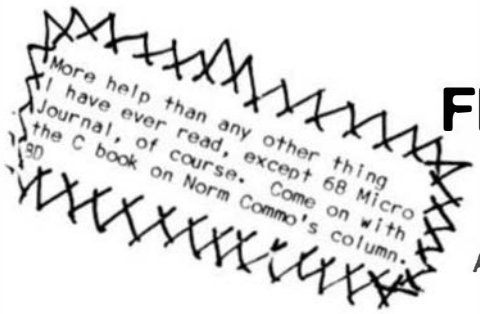
Shipping and Handling; $3.50 per Book, $2.50 per Disk Set
* All Currency in U.S. Dollars Foreign Orders Add $4.50 S/H
If paying by check - Please allow 4-6 weeks delivery

Continually Updated In 68 Micro Journal Monthly

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN. 37343

(615) 842-4600
Telex 558 414 PVT BTH

TM - OS9 is a trademark of Microware Systems Corp. and Motorola Inc.
TM - 68' Micro Journal is a trademark of Computer Publishing Inc.



FLEX™ USER NOTES
THE 6800-6809 BOOK

By: Ronald W. Anderson
As published in 68 MICRO JOURNAL™

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

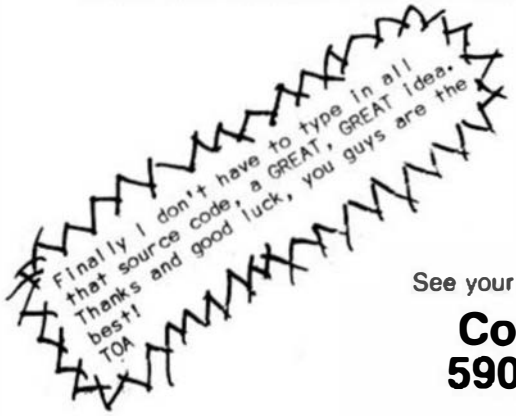
All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC
	(And many more)

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1, .C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.



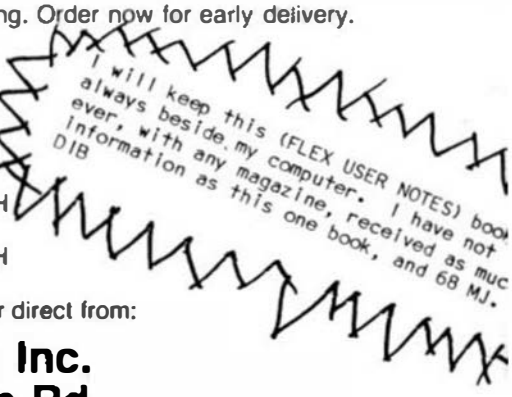
Foreign Orders Add $4.50 S/H

Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H



See your local S50 dealer/bookstore or order direct from:

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

TELEX 558 414 PVT BTH

™FLEX is a trademark of Technical Systems Consultants

FLEX User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, Mi 48105

What Makes Computing Fun?

I've been thinking a lot about that question lately. I started my "affair" with computers about 8 years ago now. What makes it such a captivating hobby and fascinating career? Hobby computing is ALWAYS fun because you only do it when you feel like it. Does that make sense? Sometimes I get very excited about some new project in connection with my hobby computing, and I'll spend a couple weeks of 2 A.M.s working on that project. Other times, I just want to get away from it for a few days, and I might spend a couple evenings writing letters to my printer pals around the country. I think what makes me ALWAYS come back to it is that each project is new.

Way back, I got interested in binary math packages, just to choose one example. I wrote a package that used the stack a lot. This was before the 6809 and of course the 6800 has only a single stack pointer, S. If you've ever tried to use it, you quickly learned that subroutines are a pain. Push some data on the stack and JSR to handle it, and the return address is in the way. Every subroutine has to pull the return address off the stack, do its function with the data, either altering it or removing it and pushing results, then either push the return address and RTS or stuff the return address in X and JMP 0,X. The math routines worked better and more efficiently than some I had just bought, and I used them for a few projects. Though it was a tough project, I gained some excellent experience in stack manipulations. When the '09 came along, I had the very interesting project of translating the math routines. A half a page of stack manipulations was reduced to three lines of code. The MUL instruction of the 6809 allowed a new approach to the multiply routine. It became more complex, more code, but it ran 4 or 5 times faster.

Later I became interested in improving some simple and rather crude approximations that I had put together early in my computing experience for Sine, Cosine, Arctangent, and Square Root. That precipitated another search for improvements. Now, every time I see an article on math and scientific functions, I look for possible improvements that I can incorporate.

That is one example out of dozens that I have experienced. You might wonder if one could ever get tired of multiply routines and Cosines. I can only say that I haven't yet. Each new possibility for improvement sends me off on a week of exploration, whether it be writing routines in assembler, modeling the calculation in BASIC to see if it is more efficient or more accurate, or tailoring a package for a particular language, the fun is there all over again.

Another reason computing has been so much fun over the past 8 years is the rapid change in technology. I've gone through my original system configuration here enough times so I won't bore you with it again, but let's take a look at just a couple of changes. Way back in 1978 or 79, I bought an 8K memory board for around $200. Now I can buy 256K for under $1000! My original disk drives were a pair of 35 track SSSD units that provided 340 sectors or about 85K of data storage each. Just at the start of the rapid advances in disk technology, I bought a pair of 8" drives of the Double Sided Double Density type, that provided nearly 1 Megabyte of storage each! That is an increase of more than 12 times. I could take 12 of my old disks, all full, and fit their contents on ONE disk. The latest technology in five inch disks, 80 tracks, double sided, double density allows about an 8 to 1 increase in storage density over those original drives. I don't know if the newest quad density drives are available in 80 track, but if they are, the storage would exceed the 8" DSDD drives that I have by 40%!

The original pair of drives and interface cost about $1000. The latest described above can be put two to a box with power supply for about $450. Now look at a 10 Mbyte hard disk WITH CONTROLLER for an IBM. These can be had for less than $700. Applications that could be only dreamed of a few years ago are now a reality. Am I still excited about computing? What do you think?

A Trend

I couldn't help putting in this little bit about the IBM and its clones. I presently have very mixed feelings about the '09 vs the IMB clones, and these comments will reflect those feelings. First point is that I have just scanned two issues of PC World looking for an Assembler for the 8086. In those two issues I found ONE! Most of the software suppliers list what they call Utility programs, but their idea of utilities is something like "Sidekick", which is a program that lets you interrupt what you are doing to make a note to yourself in a file or access a data file, and then go back to what you are doing. Obviously, the "trend" is for the computer to be a "user" item and not a "programmer" item. There are several "languages" for sale in the ads, mostly C, Pascal, Fortran, and BASIC compilers.

The company has bought the Lattice 'C' compiler for the Tandy IBM clone, and it is a rather nice compiler with terrible documentation. I managed to get my multi test benchmark running in Lattice 'C'. At this point I don't know whether to be impressed or not. McCosh 'C' on the 6809 beats Lattice 'C' on the Tandy, but not by very much. The results are quite close, the Tandy winning some of the tests and the '09 some others. I note with some satisfaction that the McCosh version on the '09 generates 11.3K of object code while the Lattice version generates 18.8K on the Tandy. However, when the Tandy is aided by the 8087 Co-processor, things are quite different. The two long times in the benchmark are those for the square root of 1000.0 taken 1000 times, and the DOUBLE precision multiply divide test in which there are 2000 multiplies and 2000 divides. The 8086 and the 6809 both take around 13 seconds for each of these tests. With the co-processor, however, the Tandy does these two in around 2 seconds each.

I bought one of the inexpensive screen editors for the Tandy, and I can truly say that it is awful. Not only won't it "wrap" a word around to the start of the next line when a line fills, it actually wraps to the start of the same line and starts to overtype what was just typed in! Further, at the bottom of the screen it wraps around to the top. The user must explicitly page down to provide more space in which to edit. If that were not enough the user also has to "extend" the file with blank lines before he can edit below the current last line. After using the editor for a while, I've gotten sort of used to its non-wrap and all the other features are rather nice.

JUST in C

No, that's not a typo. I've mentioned my text formatter program JUST in this column, and you have seen the ads for it nearby. It is written in PL/9. Well, I happen to like JUST a great deal. Having written it for myself, it just about fits my needs, which include simplicity of use and a limited number of commands. PL/9 compiles about 4.3K of object code for it. A couple weeks ago, I decided to try coding it in "C" so I could use it on the Tandy and the SWTPc system. Surprisingly it took only a week of evenings to get it translated and running. Since I wanted to get some experience with the feel of the Tandy keyboard and the new screen editor, I thought I would first get it running in Lattice "C" on the Tandy. Mission accomplished. The C version is quite fast in operation, can output to the printer, the terminal or a disk file with embedded control characters for the printer. The object code came to 11.8K, almost three times as large as the PL/9 version on the SWTPc aystem.

After getting it running, I downloaded it back to the SWTPc system and compiled it in McCosh (Windrush) "C". There were remarkably few problems of compatibility, and most were concerned with the switching of output between a file, the terminal, or the printer. When I had both versions running, I found that the difference between the two versions was all contained in 5 lines out of the some 650 lines of program. I'm impressed with "C" and its implementors.

The idea of having some software common to both systems was so nice that I immediately reactivated my screen editor project. I'm writing a screen editor that in my opinion puts together the best features of all the screen editors I have used. I decided to do it in PL/9 since it will compile about 12 times faster than with any of the C compilers, and it will be

easily translated when finished. I had about 1/4 of it done some time ago but had set it aside as a long and difficult project. Presently I am about half way toward having it done. I was right about it being difficult. Progress has been slow. My approach is to implement another command or a new feature by writing one or more new procedures for the program, getting them to compile, and then debugging them until they are somewhere near bug free. When I get tired of debugging, I write another chunk of code to implement another command, type it in, get it to compile and start debugging again. When it is done and operating respectably, I'll send a couple copies off to people like Dan Farnsworth, Don Williams, Frank Hoffman, Peter Stark and a few others for comments and bug reports. Maybe some day it will be on the market.

Feedback

I just received my June '68' Micro Journal today, and in the same mail were two reader responses. One was largely informational and asked a couple of specific questions that I can answer with a letter. The other expressed a liking for the bit on the FLEX OUTCH and OUTCH2 character output routines and output switching. It is too early to draw any conclusions from the initial response. If the timing of these is any indication, I will probably get a number of responses. The writer of one of the letters said that he is now 48 and doesn't want to sit down and learn "from the beginning", but wants to get on with the programming. Surprise, folks, my 50th birthday came last November. I guess I do like to dig into something and learn it from the basics up. I agree with something Don Williams said in his "Random Thoughts" in the June issue... We "old dogs" can learn new tricks, but he is right, it does take longer than it used to. More on the feedback later as more responses arrive in my mail.

FLEX FILES

In keeping with my promise of a few months ago, I thought I'd pick a FLEX topic for some discussion here. Flex uses two kinds of files, One is called a Binary file and the other a Text file (or an ASCII file). These designations refer to the contents of the file and how it is structured, and not the filename extensions .TXT and .BIN, though those extensions are usually used by FLEX users to distinguish between the types of files. A Binary file usually contains "object code" that is "executable" by the computer. Most programmers use the extension .BIN for a binary file, and the TSC Assembler uses that as the default extension for an assembler output file. All the FLEX utilities with the extension .CMD are binary files. Most Flex users use the .TXT extension for text files such as letters, manuals, etc. that are to be processed by a word processor to produce printed output. I tend to use the extension .TXT for all text files, but many other programmers use .SRC for Assembler source text files, .PL9 for PL/9 source files, etc. I do use .BAS for BASIC program files (which are text files), and if I used the "compiler" in TSC BASIC, I would use .BAC for "compiled" programs.

What is the difference between these two types of files? A text file contains only printable ASCII characters (with a few exceptions). Text files in FLEX use a technique called "space compression". When FLEX reads a text out of memory and writes a file, it looks for two or more successive space characters ($20). It counts the successive spaces and replaces them with a special Horizontal Tab character ($09), followed by a count of the number of spaces in binary form. For example, if there are 5 spaces in sequence, they will be replaced by $09,$05, thus coding the 5 spaces into two bytes. This system greatly reduces the size of text files with tables, etc. that are to be processed by text formatters. This space compression is "transparent" to the user of the files. That is, the file is "expanded" again when it is loaded back into memory by a program. All the HT characters and counts go away and the spaces are again inserted. A CR ($0D) is allowed in a text file. FLEX also allows nulls ($00) and $18 (control Z) which some software uses as an end of file marker in text files. Nulls are ignored by FLEX and if present in a file being read, they are "skipped over".

A text file most frequently is composed of lines of text. A standard FLEX text file uses only CR ($0D) to mark the end of each line. The LIST utility adds an LF to the end of each line when it lists a text file. Word processors must do the same.

A binary file may contain any and all possible values that can be represented with 8 bit "bytes". If you stop and think about it, that is a little trick. If the file can contain all possible values, how does the system put "instructions" in the file, such as at which address to load the

file into memory? The answer is "very carefully". A binary file is a bit more complicated than the text or ASCII file. The first byte of the file must be $02, and in fact that is the simplest way of determining whether the file is text or binary. The $02 is followed by two bytes that are the load address of the code that follows. Next comes a single byte that is a byte count for the "block" of code that follows. Since the count is a single byte, a block can't be longer than 255 bytes. Most programs are longer than that, so what happens when the byte count is exhausted? FLEX expects to see another $02, another load address, and another byte count. The process is repeated until there is no more file. If there is a "transfer address" (a transfer address is the address where the code is to start executing after it is loaded into memory), a code $16 follows the last block. After the $16 comes the transfer address. Any remaining bytes in the last disk sector containing the program are nulls ($00).

Now you may well ask how FLEX knows which kind of file to write. FLEX, as I said above, automatically processes space compression for text files. It does not automatically provide byte counts and load addresses for binary files, however. The program that loads or saves a binary file must do that. Of course the flex utilities for loading a binary file (GET) and for saving a binary file (SAVE and SAVE.LOW) read and write binary files properly. FLEX must be told, however, whether or not to use space compression. Without going into a lot of detail, FLEX uses a File Control Block (FCB) for each file that is open. When you open a file (or when one of your programs does so) FLEX defaults to a text file and performs space compression or expansion unless you set a "FLAG" in the FCB to tell it not to perform compression. If you point the X register of the '09 at the FCB (LDX # FCB1) then the byte in question is located at 59,X. (LDA #$FF, STA 59,X) will set the flag for binary files. Any program that manipulates binary files must set this FLAG after the file is opened and before it is used for read or write.

While the binary file may appear complex, it is not terribly difficult to write a program to write or read a binary file. TSC did a very good job of making the storage of binary files efficient. Four control bytes for each block of up to 255 bytes is a very small overhead compared to the way in which many other systems store binary files. Some of Motorola's older systems and many relocatable assemblers generate files called ASCII binary files. Each byte is stored as the ASCII representation of its two digit hex value. For example a $12 is stored as ascii 1 and ascii 2 ($31,$32). That of course means that the overhead is at least 100%. Most of the schemes that use ascii binary files have further overhead. The standard Motorola format some time ago for binary files on cassette tape was what is called the S1-S9 format. Each line of 16 bytes was preceded by S1, a two digit byte count, a four digit load address. Then came the 16 bytes using 32 characters, and then two more characters at the end that represent a "checksum" for that line. That made it necessary to use 42 bytes to represent 16, an overhead of almost 200%. Clearly, the little bit of coding of byte counts and load addresses were worthwhile, particularly at a time when the disk held 85K of data.

If I remember correctly, the old MDOS of Motorola called the ascii representation of the binary file a BIN file, and the more compact pure binary file an EX file. There were two utilities called EXBIN and BINEX that made the conversion. The assembler produced output in the ascii binary version and one of the utilities had to be run to translate it into a plain binary file.

There are a couple of points worth mentioning. First a text file has no load address. It doesn't matter where in memory the text file is loaded since it is not executable code. A text editor may have a buffer in one area of memory, and a word processor may use an entirely different area for a working buffer. Both can load the same text file. A Binary file may be a program written in position independent code or it may not. If it is position independent, it may be loaded anywhere in memory and run. Such programs usually require a special loader that will load them at an address specified at load time, and jump to the first load address for execution.

That is the lesson on FLEX for this time. I hate to sound like a broken record but all this information is in the Advanced Programmer's Guide that comes with FLEX also. I think, however, that I have expanded the information and given some reasons why it was done the way it was. If you like this monthly discussion of some of the more basic things in FLEX, please let me know and we will continue it for a while.

OS-9 User Notes

I have been living at a new address for almost a year. The post office has been forwarding mail to me, but they are getting tired of it. I don't live at 517 Goler House any more. My new address is:

Peter Dibble
19 Fountain Street
Rochester, NY 14620

NOTE

This month's column is going to set a record for brevity. I am feeling the pain that is part of graduate studies very clearly this month.

Part of the initiation process at the University of Rochester Computer Science Department is a form of torture called qualifying exams. They are a series of exams covering all the areas of Computer Science. I have to pass them before I can take the next step toward a PhD. I'm in the middle of them; writing this column counts as relaxation.

The labs at school are filling with 68x based computers. The largest is (of course) the Butterfly. We're hoping that by the end of the year we'll have a baby Butterfly and a big Butterfly with 128 68020s between them.

We also have gaggles of Suns. They are personal work stations with a 68010, a few megabytes of memory, a hard disk, a big screen, and an ethernet connection. They run BSD Unix. It took us a while to learn how nice they are, but I just read a note on the electronic bulletin board that threatened to set up a formal system for scheduling Suns if we couldn't do it informally. That's a sure sign of their popularity.

Personal work station is an interesting phrase. I think it means expensive personal computer. The main difference is in the history of their development. At one time a group of engineers would share a mini computer or a work station for design work. Personal work stations are computers with enough resources to do the class of work that used to be done by those shared computers but with a low enough price to be reasonable part of an individual's office equipment. It's interesting to note that most personal work stations seem to be built around 68010 microprocessors.

I haven't had much to say about school. I waited until the middle of qualifying exams before I reported on it. In the spirit of a good reviewer, I wanted to see the thing at its worst before I made critical remarks.

Here's my report: "Come on in! The water's fine." We have a wonderful collection of toys, but the thing that really makes it worth kicking a big hole in my career to go back to school is the people.

The Module Directory

The storage of modules in OS-9 Level Two is one of the most important differences between Levels One and Two. Under OS-9 Level One modules a simply read into memory. The only aspect of module storage that needs to be kept in mind is that each module takes an integral number of memory pages. Level Two does away with that restriction, but the underlying mechanism is complicated.

The entries in an OS-9 Level Two module directory are much like Level One entries. One field, the DAT (Dynamic Address Translator) image pointer, makes all the difference. DAT images for

Meet people making it happen in OS-9. The movers and shakers who are helping OS-9 become the fastest growing operating system for the 6809 & 68000 in the world.
Lively and informative round-table discussions will cover the design and use of Microware Software. We'll also discuss OS-9's dynamic growth from where we are today to where we may be in the future.
The exhibit area will feature booths from many of the leading suppliers of OS-9 compatible hardware and software. It's a great opportunity to increase your skill and knowledge in the latest microcomputer software technology. Plan to attend — Register Today!

Seminar only $150 Hotel Package* $350
Location Marriott Hotel, Des Moines, IA
Don't Miss It — Pre-Register Now!
Call 515-224-1929 or Write
MICROWARE SYSTEMS CORPORATION
1866 N.W. 114th St. • Des Moines, IA 50322

microware®

Hotel package includes 3 nights, single occupancy at the Marriott Hotel and registration fee.
OS-9 and BASIC09 are trademarks of Microware and Motorola

modules are special things.

OS-9 Level Two builds a DAT image for each process. They are attached to the process descriptor. Each time a process is dispatched, the process's DAT image is loaded into the DAT hardware. That is how every process is given its own address space.

There are two ways of finding a location in memory. Either give its real address (24 bits long) and use the service requests that handle extended addresses, or give a DAT image and an offset in the address space defined by that DAT image.

Microware could have decided to save the locations of modules as 24-bit addresses. That would have made the Level Two module storage mechanism seem a lot like the Level One system. It seems like that would be simple, but there is a problem: In order to keep things reasonably simple, a module would have to be stored in contiguous storage. That way the Module Directory could specify the location of the module by a start address and a length. Early users of Level Two will remember waiting eagerly for the release that supported discontinuous memory for modules.

The problem was that long blocks of empty memory are hard to find in a hard working system. It usually isn't an issue; with the DAT hardware, chunks of memory can be assembled from all over to build a new address space. Sometimes a program couldn't be started because it couldn't find enough contiguous memory to load a module. That problem was supposed to have been left behind in Level One. I don't know, but I bet the early Level Two versions kept module locations as a start address and a length in the Module Directory.

I have to submit this column now. I'll continue with modules in another column, but I won't keep you in suspense. The solution to the module problem for Level Two involves creating special address spaces just to hold modules.

Hot Scoops

The OS-9 Users Group software library has grown into a very substantial collection. I hear that there are about

35 disks in it. Lots of interesting programs (including XLisp).
***

I heard from Don Williams just moments ago. My collected user notes are not only being published in this country as of about now, they are also going to be published in Japan -- in Japanese! I hope I won't have to read proofs for that edition. My brother, who has been studying Japanese hard for years, tells me that it is a difficult language to learn.

- - -

Editor's Note: First, about the collection of disk for the users group. A couple of years ago, or so, we sent someone in the group (they had called after we had paid our dues) and suggested I send some disk (5 inch) and they would get the collection to us. Dunno what happened but not only did we not get the disk but we don't even get the newsletter. The only thing we have EVER gotten from the users group is a couple of 'releases' from Dale, who was president at the time. Which we promptly ran - unedited! Since then I have bought this up with an officer of the group and was told, "havn't ever gotten mine either."

The reason I go into this, in this type forum is that I hope to stir someone. OS-9 is a fine operating system, and it deserves better. I guess the problem is that like many volunteer organizations things move at a more casual pace. Also there have been changes of the guard and lost paperwork, etc. All of which tend to contribute to the confusion. However, I get newsletters, clippings, notes, disk with source programs and all sort of other media stuff, all without asking. But getting something out of the OS-9 users group is a pain, to say the least. Also I was told at one time that we (68 Micro Journal) had 'peeved' a couple of the officers. Well, how bout that. If you can't do the job, then don't make waves and over extend yourself. You will only make it worse for the other fellows, who sweat to make it a go. Like I said a few lines back - OS-9 is too good a product to have black marks because of some folks who can't or won't do it right.

The other side of the coin is that probably 99 percent of the rest of the members are humping to make it what it should be. A pity that a few loud mouths can smut up a good thing!

Peter Dibble has done yeoman effort in his getting OS-9 material to us each month. We (I especially) know what he has and still is, going through. If I were Microware I would certainly make Peter a "Knight of the CPU' or some other good and respected award. He has put his sweat where some of the others have only slide their mouths by. He has put up, rather than shut up. And as for us, Peter Dibble is TOP DRAWER! If it were not for guys like Peter Dibble, Ron Anderson, Phil Lucido, Bud Pass, Bob Nay, Ron Voights, Carl Mann, Norm Commo, Theo Elbert, and everyone of the other 2,041 who have been published in 68 Micro Journal, we would not have made it. And that also goes for a lot of those who support us with their advertising. Together we have ALL made had it work. And I personally am glad it went and goes as it does. We use the fool out of OS-9 here at CPI and we know what a fine system it is. So it really hasn't made much difference to us, we get along just fine with or without the users group. But for those who need the help a real users group can provide (check into the "C" users group sometimes, or others) I feel something has gone bongs. How about it fellows? - oops, sorry, and GALS?

Also we are glad that Peters published notes are to be published in Japan. I will have our version ready for sale by the time you read this (I hope). I let the rights to Japan this week and Peter will receive the much needed financial rewards from this that he so well deserves. OS-9 is a very popular system in Japan. Who knows, when the Japs take over; computer market made stale by everyone here wanting to look like 'Big Blue" thus killing any of the inventive efforts that once marked the USA as the leaders in the micro field. When they do the computer market like they did the electronic, auto, camera, watch, etc. markets, who knows - OS-9 might replace M-DOS. In Japan Big Blue ain't all that big!

Anyway, congratulations Peter, from all of us here, we are mighty proud to be associated with an internationally famous and published (soon to be) author. Just don't forget us. Now about that new publishing thing that can make a pile, well,

now that you .....

DMW

"C" User Notes

Edgar M. (Bud) Pass, Ph.D.
1454 Latta Lane
Conyers, Ga 30207

INTRODUCTION

This chapter discusses several of the major current implementations of C compilers on the 6809 and the debugging of C programs on any computer system.

VERSIONS OF C

There are many implementations of C on the 6809, 68000, 8080, 8088, Z-80, and other microprocessors. Like most other products, the completeness and usability of the implementations vary considerably. However, there are several major groups into which many of the implementations fall. The most complete compilers are advertised as Full-C or UNIX-like, and claim to implement K & R's C almost entirely. The least complete compilers are advertised as "Small-C" and implement only specific subsets of K & R's C. Many C compilers are complete in some areas, such as verbs, but are incomplete in some areas, such as floating-point variable types. Many C compilers claim to be complete implementations of K & R's C, but actually provide far fewer facilities.

Essentially all of the non-Full C compilers were based on or derived from Ron Cain's Small C, published in Doctor Dobb's Journal 45, which was written for the 8080. It has the advantage of being written in its own language, thus facilitating porting to new implementations. It has the disadvantage of having several major bugs, most of which were also ported to the new implementations. Most of the Small C compilers developed for the 6809 are no longer marketed, with the major exception of Dyna-C and Dugger's C.

PROGRAM GENERATION

All current versions of C for the 6809 generate some form of assembler language as an intermediate or final output. This code file is combined with other source and object modules and is compiled to produce an object module. This object module is then combined with other object modules to produce a final executable program. The details by which these steps are accomplished vary significantly among the implementations.

Several of the Small C's generate assembler code in standard Motorola format, thus avoiding the necessity of developing a special assembler, linker, or other support programs. Under FLEX, all the Small C's target the TSC assembler except for Everhart's, which targets the SWTPC assembler. However, the entire C library must either be assembled with the user program or pre-assembled and pre-loaded before the user program may be run.

The Wordsworth C compilers use the RLOAD linking program and several special macros for the TSC assembler to avoid some of these problems. RLOAD uses special external references generated with the object program to indicate which link address need to be established in order to bind a program together. It requires a separate file to indicate which files are to be included in the linking process. The C library may be included as one unit or it may be included as separate modules to save memory space and load time. In the latter case, every library routine called by the program or by another library routine used by the program must be manually determined and listed in the control file.

The Introl C and Microware/Tandy/Windrush/McCosh compilers use their own special assembler and linking loaders. With one command, the

user can produce an object module (non-executable) from a source C program. With a separate command, the user can produce an executable object program from one or more C object modules (exactly one of which is a main program) and all called libraries. They use a tree-structured approach by linking to "main", which can link to other modules, which can themselves link to other modules, etc. In effect, they automatically construct (and are able to optionally output) the table required by Wordsworth's RLOAD to be manually constructed and input by the user.

CODE GENERATION

In terms of the Full C compilers, both Introl and McCosh C compilers provide optimization. It is integral to the Introl compiler, whereas it is provided as an external program in the McCosh compiler, although it may be invoked from the command line. The McCosh optimizer is claimed to save about 11% of the code and time required to process the object code, when compared to the original code.

Since most of the Small C compilers are direct descendants of Ron Cain's Small C compiler, the object code they produce causes a 6809 to attempt to emulate an 8080. Of course, the code generated by this process is extremely inefficient, producing such code as the following:

```
* int i;
* i=0;
    TFR S,D
    PSHS A,B
    LDD #0
    STD ,S++
```

whereas Introl C would produce the following code for the same program fragment:

```
    CLRA
    CLRB
    STD 0,S
```

which is probably similar to what most assembly programmers would have written to accomplish the same operation.

Rather than correct the compiler to emit better code, Wordsworth and Everhart C compilers provide optimization programs (written in C) to scan the assembler file output from the compiler and correct the most common and blatant sequences of inefficient code.

The Wordsworth optimizer is a peephole optimizer, utilizing a 4-line peephole. It processes the following sequences:

```
OPTIMIZED       ORIGINAL

TFR X,D         (Removed)
TFR D,X

TFR X,D         PSHS X
PSHS A,B

LEAX p,S        LDD p,S
TFR X,D
TFR D,X
LDD 0,X

TFR X,D         PSHS X
PSHS A,B
TFR D,X

TFR S,D         TFR S,X
TFR D,X
```

When these substitutions are made, Wordsworth claims that the result should take up from 65% to 85% of the space required for the Small C compiler output.

The Everhart optimizer is also a peephole optimizer, although it has a variable size peephole. It processes the following sequences:

```
OPTIMIZED       ORIGINAL

LEAS -2,S       LEAS -n,S
:
LEAS -2,S

LEAY p,S        LDD p,S
TFR Y,D
PSHS D
LDD [,S++]

LEAY p,S        LDB p,S
TFR Y,D
PSHS D
LDB [,S++]
```

Everhart claims that the result should be 15-20% smaller than the original after these substitutions have been made.

Wordsworth's Middle C, Everhart's Small C, and Intersoft Small C (version 2) cleaned up some of the worst code generated by

earlier versions of C. Optimizers are still needed, however. Dugger's Small C compilers never generated code quite as inefficient as Intersoft Small C version 1. However, they have many other problems in terms of generating incorrect code, which is far worse than inefficient code.

All of the C compilers reviewed here, except for Introl C, have the option for placing the C source program as comments in the assembler source program. This is very convenient for documenting the generated assembler program and for assisting in debugging the C program, which leads into the next section of this chapter.

DEBUGGING C PROGRAMS

Currently, debugging a complex C program on most systems can be quite an adventure. Following are described several possible means of assisting in debugging C programs, in order of increasing sophistication, and their current practical uses and limitations.

"Compiler Syntax Checking" is provided automatically by all C compilers. It warns the user of many syntactically incorrect conditions. It usually does not warn the user of such conditions as incorrect number or type of parameters of a call to a function and several other serious conditions. The C language avoids many of the problems of FORTRAN and BASIC languages by requiring the declaration of all variables and typed functions before use. However, it allows the creation of new categories of problems thru the use and misuse of pointers. C's structured constructs (WHILE, SWITCH, DO, FOR, IF, etc.) assist the user in structuring programs, and the C compiler can detect many forms of badly-structured constructs.

"Compiler Run-Time Checking" is also provided automatically by all C compilers in their run-time packages, although the level of checking provided varies significantly among compilers. All check for common conditions such as attempting to read a non-existent file or an I/O error occurring during a read or write operation. However, very few of them validate the file pointer passed to the I/O routines. Both "fprintf" and "fputs" require file pointers; however, "fprintf" requires the file pointer as its first parameter and "fputs" requires the file pointer as its last parameter. Inconsistencies such as this can trap the unwary user, with potentially disastrous results. Another major type of run-time problem not detected by most C compilers is the "subscript out of range" situation, which is flagged by almost all BASIC interpreters. This problem can easily cause destruction of program, data, and operating system routines, with disastrous results.

"Desk-Checking" is debugging performed by the user without the direct assistance of the computer. In earlier times of computing, when computer time was extremely expensive compared to human time, desk-checking was heavily emphasized. In more modern times of computing, desk-checking has been de-emphasized. There are still good uses for desk-checking, however, such as the case in which several people share one terminal or computer or the case in which the computer is available only certain hours of the day or certain days of the week.

"Program-Verifier Program" extend the syntactic and semantic checking of the C compiler in certain directions. The program "lint" on UNIX systems attempts to detect features of C programs which may be bugs, non-portable, or inefficient. It checks type useage and implicit type conversion very carefully. It also detects unreachable statements, loops entered at other than the top, variables and functions declared but unused, variables used but not initialized, and logical expressions with constant values. It checks for functions returning values in some cases but not others, functions called with different numbers or types of arguments than the declaration, those with unused values, and those used with values but declared without them. The program "cchk" on micros provides a fast pre-check for C programs to locate errors that the C compilers do not specifically attempt to locate. It checks all types of brackets for correct matching and nesting. It checks indentation, which it uses to provide checks on bracket nesting. It checks for dangling "else"s, for the use of "=" rather than "==" in a conditional expression, and for nested comments. When such programs are available, they can be used to great advantage to assist in degugging C programs.

"User-Inserted Debugging Code" is used in C programs, just as in other languages. The C preprocessor facility can be used to great advantage to provide various levels of debugging code permanently in a source program without affecting the production version of the program. As required by later debugging or maintenance, the debugging code may be selectively re-activated. This debugging code may be either passive or active. Passive code usually takes the form of "printf" or similar statements to output location codes and values of indicative variables. These outputs may be unconditional or based upon various conditions in the program. Active code can take many forms, but usually takes the form of code which checks for unacceptable conditions and takes corrective action after reporting the problem, so that program execution and debugging may continue. An example of active debugging code would be a function which checks a subscript for being within a specified range for a table; if it is, the function returns the subscript value unchanged; if it is not, the function prints an error message and returns either zero or the maximum value allowed. Used properly (and sparingly), some basic debugging code could be left permanently in a program to assist in finding operational problems.

"Debugging Programs" may be used to assist the user in locating problems in the object program produced by assembling (and perhaps linking) the C program and libraries. The TSC DEBUG Program for FLEX is a good example of programs providing the features required to do an adequate job of debugging object programs. It allows the simulation of programs written for the host system (some allow the simulation of programs written for other systems, such as for dedicated single-board computers). This simulation may be performed in single-step, step-until-trapped, or in several other modes of operation. It allows memory inspection and modification in hexadecimal, hexadecimal and alpha, or instruction formats. It allows the establishment of breakpoints to trap real-time execution of program code, as opposed to simulated execution. It allows the protection of memory (during simulation) in execute-protect, write-protect, memory-protect, and simulate-protect modes. It provides a simple hexadecimal calculator for address and other calculations required by the user during the simulation. It provides transfer and stack nest level traps to help detect bad changes of control and bad stack handling. It provides a traceback of the previous 255 simulated instructions. It provides a means of simulating interrupts. It provides other capabilities designed to enhance its use, such as the ability to issue FLEX commands. Its primary problem is that it is an absolute debugger, and the user must know the absolute addresses of all routines and variables of interest in the C program. When object modules are linked together, their addresses will depend upon the preceding modules included in the link process. Several of the full-C compilers provide a link-edit map, as an optional output, providing the names, starting addresses and lengths of each module comprising an object program. If this map is not available, the user must have an assembled listing of each module and perform the hex arithmetic required to relocate each module. With some experience in using a good debugger, the user can become quite proficient.

"Symbolic Debugging Programs" have most or all of the capabilities of normal dubugging programs plus the ability to use the files generated by the assembler and linker describing the locations, types, and lengths of all program variables and program modules. Then, most of the arithmetic required to be done by the user to relocate modules and variables may be done automatically, and the debugging process is much more similar to debugging a program using a BASIC interpreter. The user is able to inspect and modify the values of variables by their names from the original C program, a very powerful debugging tool. Symbolic debuggers for C programs are available for UNIX, BDS, and other versions of C, but are not yet available for any 6809 C compilers.

SUMMARY

This chapter discussed most of the major implementations of C available for the 6809. It summarized how the compilers generate object modules and how programmers debug the programs produced by these C compilers.

68000 User Notes

Phillip Lucido
2320 Saratoga Drive
Sharpsville, Pa 16150

It's HERE HERE HERE!

If you've been faithfully following this column lately, then you know that I have been waiting (rather gracelessly) for my copy of **Inside Macintosh**. This is the manual which is an absolute requirement for any programming on the Mac (I'm talking about writing stand-alone applications here, not programs running under Basic or Pascal interpreters). Well, after about two months, it has arrived, with many sighs of relief from yours truly.

First, the physical characteristics. This book is a monster! It is 1450 pages long, divided into no less than 33 chapters. At first glance, it appears to be a phone book for a decent sized city, of perhaps a couple of million people. The resemblance continues when you open the book, since the paper used is similar to the thin stock used in a phone book.

The book is labelled a 'Promotional Edition'. A note on the inside front cover explains that, while plans have been made with "a major publisher" for a final edition to be sold "at better bookstores everywhere by late summer '85," this edition was made available in the interim, to handle the current demand. This edition is thus preliminary and subject to change.

For all of its length, **Inside Macintosh** is still densely packed with information. It is organized into a number of separate chapters, with each explaining a particular aspect of Macintosh programming. These include using the Quickdraw graphics routines, controlling the menu selection process with the Menu Manager, and handling windows with the Window Manager. Each chapter is well organized, starting with a brief description of the chapter's subject, and slowly progressing to more and more detailed information, followed by tables and glossaries quickly summarizing the entire chapter. There is also an index for the entire manual.

If you're interested in writing stand-alone applications or desk accessories for the Mac, get this manual. It covers just about every aspect of software programming you might ever encounter, in perhaps greater depth than you might like. I was nearly overwhelmed at first when confronted with this thick compendium of knowledge. I ended up turning off my Mac, sitting down, and reading the manual straight through. This took about two weeks of free time, but was well worth the understanding it brought. A useful hint - I found it handy to put index tabs at the start of each chapter, so I could quickly dig my way through the manual. This becomes necessary when you start searching for information while programming, since this often seems to involve cross referencing back and forth through various sections of the manual.

Once I've read IM, a number of things become clear. First, Apple has done a very nice job in writing the ROM used in the Macintosh, and in defining an environment for future programming of applications. A lot of thought has obviously gone into the machine, especially as regards the 'user interface', the manner in which the machine communicates with the user. The various Macintosh concepts, such as windows, dialog and alert boxes, pull-down menus, etc., are actually all predefined functions, built into the ROM. For the most part, a program must

simply recognize when a certain action is to be taken, like the mouse button being pressed inside the menu bar, indicating a menu selection, and call the ROM routine which performs the specific action. Just about everything is already done for the programmer, whose only job is to provide the glue between these various ROM routines.

Another point which becomes obvious on reading IM, though, is that a lot of glue is required. There are about 500 different built-in routines (system traps) provided in the Macintosh and described in the manual. Programming is not going to be easy, since so many trap functions will need to be used. As an example, consider a minimal program you might write in C when starting to produce useful code. No more than a handful of routines, such as **printf**, **strcpy**, and **getc**, need to be learned before you can produce a program which is actually capable of any normal complexity. For the Macintosh, though, you need to use at least 25 different trap functions just to implement a minimum program with windows, simple text-handling or graphics, and such standard Macintosh concepts like menus.

So how am I to cover the Macintosh? Frankly, I'm a little puzzled. There is too much introductory material for a small column like this to cover in any short period of time. There is just no substitute for actually getting your own copy of **Inside Macintosh**. There are some things I can do, though. If there is enough interest, I can present short program sections which have to do with a specific problem in Macintosh programming, like writing desk accessories or getting a menu to work. Entire programs of any complexity will be difficult, because of their length. In any case, I need your letters to guide me.

My First Program

Having said that entire programs are too long for this column, I now present a complete program for your edification. Actually, I'm not being too self-contradictory. This program is that minimal example I described earlier. All it does is put a window on the screen, in which text can be placed via the keyboard. The window can be dragged around on the screen, though its size cannot be changed. The text can be selected via the mouse, and cut and pasted via menu selections. The necessary code for using menus to select desk accessories is also present.

This program is in no way original with me. In fact, it is a simple adaptation in C of the sample Pascal program to be found in the Road Map chapter of the manual. I did this program just to get used to the procedures necessary to develop code for the Mac. It is written with extensive comments, so you can hopefully follow along without too much difficulty.

Before going through the program, some preliminary explanation is in order. A Macintosh program is not usually written as one monolithic piece of code. Instead, it is divided into "resources." These are pieces of data, describing some element of the application, such as a window or menu. By keeping these descriptions separate from the actual code, it becomes a much simpler job to modify the entire application, say for a foreign language. When the code needs to create a menu or window, it just passes a "resource ID" to a trap function, which automatically reads the resource from the disk (there is obviously much more involved here - I am just too pressed for space to go over it more deeply).

On to the program. After the initial comments come a number of **#include** statements, which read some standard header files provided with the C compiler (Aztec C in this case). These files define the various structures used in Macintosh programs, as well as the names of the built-in trap functions which are available. Following this is a definition of the manifest constant **NULL**. An aside here - in this C, integers are 16 bit quantities, while pointers are 32 bits. If I defined **NULL** as 0, then using it in a function call would result in the wrong size value being placed on the stack. Casting to a character pointer forces a 32 bit value. This is also preferable to "#define NULL 0L", which is not portable to other machines.

Next, the resource ID definitions supply the numbers of the menu resources used by this program. The various menus are defined separately and referred to by these IDs. Various definitions follow having to do with the order of menus and the order of items within the Edit menu, as well as the resource ID for the window. Finally, global variables used in the program are defined, using the standard Macintosh structure names.

Now for the code. The program starts by initializing just about everything in sight. These calls bring the various sub-systems in the ROM to a known starting point. A C function, **SetUpMenus()**, creates the menu bar. A number of routines define the characteristics of the window to be used, and enable the use of the text handling traps with the window.

Next comes the main loop. A Macintosh program is quite different from a normal program, in that it is event-driven instead of algorithm-driven. That is, a Macintosh program usually sits in a loop, waiting for the user to do something like press the mouse button or a keyboard key. When an event is detected, the proper action is taken. In contrast, most programs on other computers have a limited number of options at each step, which follow one another in a strict order. A Mac program must be prepared for anything at any time (the heart of "user-friendly" applications).

The main event loop performs actions which must be constantly taken care of, then checks to see if any event has taken place. If yes, then a switch statement chooses between the various possibilities. The most complicated of these is the mouse button being pressed. When this happens, another switch uses a trap function to see where the mouse was located at the time. If the mouse was in the menu bar, then a C function is called to do the menu selection. Inside a window created by a desk accessory, another trap is called to handle the event. If the mouse was in the drag region (the title) of this program's window, then yet another trap is used to move the window around the screen. Finally, if the mouse is in the main part of the window, the window is either selected, or if it is already active, the mouse click is used to select text within the window.

Now for the other events. A keypress is passed to a text handling trap, which takes care of storing it in memory and displaying it within the window. An activate event occurs when the program's window is activated or deactivated (by desk accessories become inactive or active, respectively) with the proper trap functions being called. Finally, an update event occurs when something that was obscuring the window is removed, so the contents of the window must be redrawn.

Next come the subroutines. **SetUpMenus()** reads the resource definitions of the menus from the disk, and adds the names of the desk accessories to the Apple menu. The menu bar is then created and drawn (two separate steps). **DoCommand()** is called whenever a mouse click occurs in the menu bar. It is passed the result of a trap function that handles the entire menu display and selection procedure, which returns a number describing the chosen menu and item within that menu. A switch statement selects between the various menus. For the Apple menu, the selected desk accessory is started. The only entry in the File menu is Quit, so a flag that will terminate the event loop is set. For the Edit menu, the standard cut and paste options are passed directly to the text handling trap functions.

That's the C program. The second listing is the input used for RMaker, the resource compiler. This text file describes the various resources used by the program. The three menu resources are defined, followed by the window resource definition. At the end, the actual code for the program is named. RMaker reads this file, creating the resources in the format expected by the Macintosh trap routines. For a complete description, read the RMaker documentation which should be available with your compiler system. RMaker is from Apple, but is normally available only as a part of a software development package.

Sorry if this description seems a bit hurried. As you can see, a Macintosh program is complicated, so even this small example pushes the limits of this column. (Also, I'm writing this at 11:30 the night before I leave on vacation - oh well.) If you want to know more, such as what the individual trap functions used are actually doing, get IM and be prepared to exert much brain sweat.

OS-9/68K Make Bug

While playing around with the new **make** utility in OS-9/68K, I came across a fairly serious bug. Fortunately, I was able to come up with a temporary fix. As the bug turns out to be something easily done in C programs, I decided to describe it here.

make uses some standard names, such as **ODIR**, **SDIR**, and **RDIR**. These can be assigned new string values, which modify the

operation of **make**. An assignment may only be done once, though, with any extra assignments ignored. When I was using **make**, I found that even the first assignment was being ignored. To further confuse the issue, I was sure that I had encountered no such problem the night before, while using the same text files.

So much for the symptoms. By using the debugger and disassembling portions of the code, I was able to determine what the problem was. The code that checks for a second assignment to the standard names does so by comparing the current name value with the null string "". If the string is null, then no assignment has been done. The standard names are simply character pointers, which should point to a zero byte (the null string) at the start of the program. That is, the variables must be initialized by **char *sdir = "";**.

Here's the bug. Instead of initializing these variables, they are uninitialized, so they start out holding the 32 bit pointer value zero, or **NULL**. Checking for a null string results in looking at the byte addressed by the pointer, which will be the byte at address $000000. The **make** program has no control over the value at this address. If it happens to be a zero, **make** will run successfully. If it is not a zero, then **make** will fail.

Now, the solution. In my computer, a program running in user state, like most procedures under OS-9, has RAM at address $000000. To force this value to zero, I wrote a short C program, **clear0**, which I placed in the startup file. Here is the entire listing of **clear0**:

```
main()
{
    *(char *)0 = 0;
}
```

End result - one bug fixed (or at least successfully masked).

- - -

Editor's Note: O.K. Phil, here is the vote for additional tutorial (desk-accessories, etc.), as you ask above. Everybody votes - YES. Thanks Phil.

DMW

- - -

```
/*
    sample.c

    A small sample application written in Aztec C.
    It displays a single, fixed-size window in which the user
    can enter and edit text.

    Adapted from the Pascal program Samp, in the Inside Macintosh
    manual (Road Map, Page 15).
*/

#include <quickdraw.h>
#include <menu.h>
#include <window.h>
#include <textedit.h>
#include <event.h>
#include <desk.h>
#include <inits.h>

#define NULL    ((char *) 0)

/* event record modifier missing from event.h */
#define activeFlag  0x0001

/* resource IDs/menu IDs for Apple, File, and Edit menus */
#define appleID     128
#define fileID      129
#define editID      130

/* index for each menu in array of menu handles */
#define appleM      0
#define fileM       1
#define editM       2

#define menuCount   3       /* total number of menus */
#define windowID    128     /* resource ID for application's window */

/* menu item numbers identifying commands in Edit menu */
#define undoCommand     1
#define cutCommand      3
#define copyCommand     4
#define pasteCommand    5
#define clearCommand    6

MenuHandle myMenus[menuCount];
Rect dragRect, txRect;
Boolean extended, doneFlag;
EventRecord myEvent;
WindowRecord wRecord;
WindowPtr myWindow, whichWindow;
TEHandle textH;

main()
{
    /* perform standard initializations */
    InitGraf(&thePort);
    InitFonts();
    FlushEvents(everyEvent,0);
    InitWindows();
    InitMenus();
    TEInit();
    InitDialogs(NULL);
    InitCursor();
    /* set up menus and menu bar */
    SetUpMenus();
    /* call QuickDraw to set dragging boundaries; ensure at */
    /* least 4 by 4 pixels will remain visible */
    SetRect(&dragRect,4,24,
            screenBits.bounds.right-4,screenBits.bounds.bottom-4);
    /* flag used to detect when Quit command is chosen */
    doneFlag = FALSE;
    /* put up application window */
    myWindow = GetNewWindow(windowID,&wRecord,-1L);
    /* call QuickDraw to set current grafPort to this window */
    SetPort(myWindow);
    /* define rectangle for text in window; call QuickDraw to */
    /* bring it in 4 pixels from left and right edges of window */
    movemem(&thePort->portRect,&txRect,sizeof(Rect));
    InsetRect(&txRect,4,0);
    /* call TextEdit to prepare for receiving text */
    textH = TENew(&txRect,&txRect);
    /* main event loop */
    while (!doneFlag) {
        /* call Desk Manager to perform any periodic actions */
        /* defined for desk accessories */
        SystemTask();
        /* call TextEdit to make vertical bar blink */
        TEIdle(textH);
        /* call Toolbox Event Manager to get the next event */
        /* that the application should handle */
```

```
    GetNextEvent(everyEvent,&myEvent);
    switch (myEvent.what) {
    /* mouse button down: call Window manager to learn where */
    case mouseDown:
        switch (FindWindow(pass(myEvent.where),&whichWindow)) {
        /* menu bar: call Menu Manager to learn which command */
        case inMenuBar:
            DoCommand(MenuSelect(pass(myEvent.where)));
            break;
        /* desk accessory window: call Desk Manager to handle */
        case inSysWindow:
            SystemClick(&myEvent,whichWindow);
            break;
        /* title bar: call Window Manager to drag */
        case inDrag:
            DragWindow(whichWindow,pass(myEvent.where),&dragRect);
            break;
        /* body of application window */
        case inContent:
            /* call Window Manager to check whether it's the */
            /* active window, and make it active if not */
            if (whichWindow != FrontWindow())
                SelectWindow(whichWindow);
            else {
                /* it's already active: call QuickDraw to */
                /* convert to window coordinates for TEClick */
                GlobalToLocal(&myEvent.where);
                /* call TextEdit to process event, passing */
                /* status of shift key */
                extended = (myEvent.modifiers & shiftKey) != 0;
                TEClick(pass(myEvent.where),extended,textH);
            }
            break;
        }
        break;
    /* key pressed: pass character to TextEdit if meant */
    /* for application window, else ignore it */
    case keyDown:
    case autoKey:
        if (myWindow == FrontWindow())
            TEKey((char) (myEvent.message & charCodeMask),textH);
        break;
    /* activate event */
    case activateEvt:
        if (myEvent.modifiers & activeFlag) {
            /* application window is becoming active: call */
            /* TextEdit to highlight selection or display */
            /* blinking vertical bar, and call Menu Manager */
            /* to disable Undo, since application doesn't */
            /* support Undo */
            TEActivate(textH);
            DisableItem(myMenus[editM],undoCommand);
        } else {
            /* application window is becoming inactive: */
            /* unhighlight selection or remove blinking */
            /* vertical bar, and enable Undo, since desk */
            /* accessory may support it */
            TEDeactivate(textH);
            EnableItem(myMenus[editM],undoCommand);
        }
        break;
    /* update event: window appearance needs updating */
    case updateEvt:
        /* call Window manager to begin update */
        BeginUpdate((WindowPtr) myEvent.message);
        /* call QuickDraw to erase text area */
        EraseRect(&thePort->portRect);
        /* call TextEdit to update the text */
        TEUpdate(&thePort->portRect,textH);
        /* call Window Manager to end update */
        EndUpdate((WindowPtr) myEvent.message);
        break;
    }
  }
}

/* Set up the menus and the menu bar */
SetUpMenus()
{
    register int i;

    /* read Apple menu from resource file */
    myMenus[appleM] = GetMenu(appleID);
    /* add desk accessory names to Apple Menu */
    AddResMenu(myMenus[appleM],'DRVR');
    /* read File menu from resource file */
    myMenus[fileM] = GetMenu(fileID);
    /* read Edit menu from resource file */
    myMenus[editM] = GetMenu(editID);
    /* install menus in menu bar and draw the menu bar */
    for (i = 0; i < menuCount; ++i)
        InsertMenu(myMenus[i],0);
    DrawMenuBar();
```

```
/* Execute command specified by mResult, the result of MenuSelect */
DoCommand(mResult)
    unsigned long mResult;
{
    char name[30];
    register short theItem;

    theItem = mResult;      /* 4 byte long into 2 byte short */
    /* process by menu ID in high-order word of mResult */
    switch ((short) (mResult >> 16)) {
    case appleID:       /* Apple menu */
        /* call menu manager to get desk accessory name, and */
        /* call Desk Manager to open accessory */
        GetItem(myMenus[appleM],theItem,name);
        OpenDeskAcc(name);
        /* call QuickDraw to restore application window as */
        /* grafPort to draw in (might have been changed */
        /* during OpenDeskAcc) */
        SetPort(myWindow);
        break;
    case fileID:        /* File menu */
        /* quit - main loop repeats until donePlag is TRUE */
        donePlag = TRUE;
        break;
    case editID:        /* Edit menu */
        /* call Desk Manager to handle editing command if */
        /* desk accessory is the active window, else perform */
        /* the proper editing command in the application */
        /* window using TextEdit */
        if (!SystemEdit(theItem - 1))
            switch (theItem) {
            case cutCommand:
                TECut(textH);
                break;
            case copyCommand:
                TECopy(textH);
                break;
            case pasteCommand:
                TEPaste(textH);
                break;
            case clearCommand:
                TEDelete(textH);
                break;
            }
        break;
    }
    /* to indicate completion of command, call Menu Manager */
    /* to unhighlight menu title (highlighted by MenuSelect */
    HiliteMenu(0);
}
```

```
* sample.r
*   Resource Compiler input file for Sample application
*   Written in Astec C, from the original Pascal version
*   in Inside Macintosh

*  Philip Lucido, May 10, 1985

junk/sample
APPLSAMP

Type MENU
  ,128 (4)
* the apple symbol
  \14

  ,129 (4)
  File
    Quit

  ,130 (4)
  Edit
    (Undo
    (-
    Cut
    Copy
    Paste
    Clear

Type WIND
  ,128 (36)
  A Sample
  50 40 300 450
  Visible NoGoAway
  4
  0

Type SAMP = STR
  ,0
Sample Version 1.0 -- May 10, 1985

INCLUDE junk/sample.code
```

ADA[R] And The 68000

By:
THEODORE F. ELBERT
THE UNIVERSITY OF WEST FLORIDA
PENSACOLA, FLORIDA 32514

PART 4 - ADA PROGRAM UNITS

In previous parts of this series, the reasons behind the development of the Ada language have been explored, and the general features of the language -- particularly those that support sound software engineering practice -- have been presented. Here, the program units of the Ada language will be considered. There are four forms of program units:

- subprograms
- packages
- tasks
- generic units

Before individually considering each of these program units, it is necessary to recognize the various classes of programming languages.

Among programming languages, five general classes can be identified. These are:

- assembly languages
- systems implementation languages
- static high order languages
- block structured languages
- dynamic high order languages.

These general classes differ in the manner in which they relate to the underlying machine, in the manner in which storage is allocated, and in the run-time support requirement.

Assembly languages are unique to the machine upon which the resulting code is to be executed. In general, there is a one-to-one relationship between assembly statements and machine language instructions, although most modern assembly languages permit the use of macro-instructions -- which are equivalent to a number of basic assembly instructions -- and of some of the modern control constructs such as IF-THEN-ELSE and basic loop structures. Storage allocation is static, in that all required data storage is allocated at assembly time. Assembly language programs do not reflect the basic fabric of the program design, as do programs written in higher level languages, so that the reliability and modifiability of the resulting software suffers. Because assembly languages are unique to the underlying hardware, assembly programs are not portable. They are, however, usually efficient.

Systems implementation languages, such as C, provide facilities such as control structures and type checking, but they preserve some of the direct access to the underlying hardware so useful in assembly languages. Statements in these languages translate to many machine code instructions.

Static high order languages provide control structures and the declaration of variables, but the underlying machine is completely transparent. This fact enhances the portability of programs written in high order languages, but forces the use of assembly language subprograms when access to the underlying machine features is required. Storage allocation is static and is made at compile time. COBOL and FORTRAN are the best known static high order languages.

Block structured languages are distinguished from static high level languages principally by their ability to provide a limited form of dynamic storage allocation called block structuring. The program is divided into program blocks -- clearly delineated areas of the program, such as subprograms. Execution of a program requires a run-time support system that allocates storage upon entry into a block, and deallocates the same storage when the block is exited. Upon entry to

each block, program execution is interrupted so that the run-time support system can allocate the required storage. Thus, the storage requirements are not known at compile time. Furthermore, since storage allocation is dynamic, variables of the program are actually created and destroyed as the program executes, creating areas of the program in which certain variables exist, and other areas in which they do not. This fact underlies the concept of the scope of a variable or other program entity. Block structured languages are also characterized by the ability to classify program objects as being of a certain type. Like the languages ALGOL and Pascal, from which it was derived, the Ada language is a block structured language.

Finally, dynamic high order languages are distinguished by the feature that all storage allocation is done dynamically. Execution of individual statements in a dynamic order level language can cause storage allocation or deallocation. The LISP language, now coming into wide use in artificial intelligence applications, is an example of a dynamic high order language.

It is important to recognize the fact that Ada is a block structured high order language, for then many of the language features become more understandable. For example, blocks in the Ada language are represented by the three basic program units -- subprograms, packages, and tasks. Each of these program units may have a declarative region in which objects requiring storage are declared. Execution of code within these blocks is preceded by dynamic allocation of the storage required for execution of the block -- or at least an implementation of the language must behave as though dynamic storage allocation takes place in this manner. The process by which a declaration achieves its effect in the Ada language is termed elaboration. It is important to note that elaboration takes place during program execution -- not at compile-time.

The basic functional entity in Ada is the program unit. An Ada program is composed of one or more of these program units. Each program unit can be independently written, compiled, tested, and stored in a program library. A program unit, in turn, normally consists of two parts: the specification -- which contains information pertinent to the interfacing of the program unit with other program units -- and the body -- which contains the implementation details in terms of executable statements. The specification and body of a subprogram or package may themselves be separately compiled, further enhancing the capability of developing a large program as a set of nearly independent software components. This feature of the Ada language is key to its ability to manage software complexity, and in turn it contributes to the reliability, modifiability, and reuseability of Ada program code.

As with other languages, the subprogram in the Ada language is normally used to implement an algorithm. There are two basic kinds of subprograms -- procedures and functions. Procedures produce a series of actions by the use of executable statements, and they may communicate with the calling environment through the use of parameters. The procedure is also the basic executable unit of an Ada program -- that is, the main program is a procedure. Functions, on the other hand, are normally used to compute a value, which is then returned to the environment under the function name. All of this sounds very much like the description of subprograms in other languages, but in the Ada language the concept of a subprogram is coupled with strong type checking and other features which tend to enhance the modifiability, reliability, and reusability of Ada software.

Additional features of Ada's subprograms include the ability to separately declare and compile the specification and the body of the subprogram. The specification constitutes the interface of the subprogram with other program units, so that once the specification is compiled other program units -- in the process of their compilation -- may reference the subprogram, even if the body of the subprogram has not yet been compiled. The body, on the other hand, is the actual implementation of the algorithm expressed by the subprogram. The body contains the executable code constituting the process implied by the algorithm. Because the interface with the calling environment is effected only by the specification, the body of a properly designed Ada subprogram may be modified and recompiled without requiring recompilation of other program units.

Packages are unique to the Ada language, although a similar feature exists in the Modula-2 language. The Ada package is used to encapsulate groups of logically related entities. The term "logically

related entities" refers to any set of program entities -- subprograms, packages, tasks, type declarations, or object declarations -- grouped according to function or application. Thus, a package may be as simple as a set of constant declarations -- such as English-to-metric conversion factors -- or it may be a set of subprograms and an associated set of type and object declaration -- such as a subprogram package designed to perform complex matrix manipulation.

Ada's packages also may have separately compilable specifications and bodies, with the specification representing the interface between the package and other program units by which it is referenced. The ability to compile the specification prior to compiling the body -- or for that matter, even prior to writing the body -- enhances the functionality of the package concept in the large system design environment, in which several programming teams may be working more or less independently on different software components of the same system. The package concept also enhances the modifiability of software, since package bodies may be modified at will and, so long as the specification remains intact, the interfacing of the package to other program units remains unchanged.

Ada's packages are also important to the concept of reusable software. They hold promise for a "black box" approach to software development, analogous to the effect that large scale integration has had on hardware development. Packages that implement certain algorithms, for example, can be marketed by specifying only the input-output relationships of the package. The actual implementation -- represented by the body of the package and analogous to the semiconductor chip in an integrated circuit package -- is of no concern to the user. Only that information contained in the specification -- analogous to the pin-out configuration of an integrated circuit package -- need be known to the user, along with a general description of the algorithm implemented. Indeed, Ada's package concept -- combined with the portability provided by other language features -- offers promise of a software industry long needed but never really provided by other languages.

Concurrent processing in sequential machines -- the process by which there is more than one related set of sequential statements being executed simultaneously, either on separate processors or time-sliced on a single processor -- is not a unique requirement of embedded systems. For many years large main-frame systems have provided multiprogramming or multiprocessing features to enhance processor utilization and to provide a multi-user environment. More recently, operating systems for microprocessor based machines have provided this capability even in computers of modest size. In the embedded system environment, concurrent processing is required not so much for processor efficiency as it is to satisfy the requirement that the system respond to stimuli from its environment in what is termed "real-time". That is, the response generated by the system to a set of input stimuli must be generated in a time frame appropriate for the response. Since there may be several -- or even many -- sets of stimuli requiring attention by the computer, and since they generally occur at random times, it is often necessary to resort to concurrent processing in order to ensure timely real-time processing of the input data. In systems using languages other than Ada, the synchronization of and communication among concurrently executing sections of code is accomplished in one of two ways. For those programs written in assembler, an executive routine of some sort -- probably interrupt driven -- is written to provide the synchronization function. For those programs written in a high order language, a run-time support package -- termed a real-time operating system -- is linked with the program and the result loaded into the computer memory. These real-time operating systems normally provide an interface with the underlying hardware at a primitive level. Primitive system functions usually include provision for input-output, synchronization of and communication among concurrently executing code segments, and management of memory resources. While the use of a real-time operating system relieves the programmer of the necessity for dealing with the underlying hardware at a primitive level, it still requires him to step out of the high order languages and into the operating system by use of operating system calls. Since these system calls are specific to a particular real-time operating system, and since the operating system themselves are often machine specific, the portability of the resulting program suffers.

Concurrent processing in the Ada language is provided by the task structure. A task is a set of instructions, together with associated type and object declarations, which is executed concurrently with other tasks -- either

through multiprocessing or through time-slicing on a single processor. Tasks, like subprograms and packages, have a specification and a body, but task specifications and bodies cannot be separately compiled. The advantage that Ada's tasks provide is the fact that task synchronization and intertask communication can be effected directly in the high order language, thus precluding the necessity for explicit interaction with the underlying operating system. Such interaction is still present in an Ada program, of course, but it is implicitly provided as a response to the execution of statements in the high order language. The process by which synchronization of and communication among concurrently executing tasks are accomplished in the Ada language is called a rendezvous. In Ada, a rendezvous is produced when certain statements are executed in the communicating tasks.

A very powerful feature of the Ada language, and one that is almost necessary as a result of the strong typing characteristics of the language, is its capability of providing generic program units. This feature permits the programmer to parameterize either a package or a subprogram in a manner that can best be explained by use of a simple example. Suppose a requirement exists in a program for a subprogram that will sort the elements of an array. The array elements may be either integers or characters. In many languages, a single sort subprogram could be used for this purpose, since the algorithm is the same in either case. In the Ada language, however, the strong typing characteristics prohibit the assignment of character values to objects of type INTEGER, or the assignment of integer values to objects of type CHARACTER. Thus, two sort programs will be required -- one for an integer array and one for a character array. A preferred alternative would be to provide a generic sort subprogram, complete in every detail except for the type of the element of the array to be sorted. A copy of the generic subprogram can be obtained, with the missing type provided, by a single Ada statement providing what is termed an instantiation of the generic subprogram. The generic subprogram may be instantiated any number of times, creating any number of copies. Furthermore, different types may be provided for each instantiation. There are restrictions on the types which can be specified at instantiation, but the utility of this feature is immediately evident.

Ada's generic units comprise generic subprograms and generic packages. They are usually considered to be a template from which corresponding subprograms or packages can be obtained through the process of instantiation. The generic unit may be parameterized, as in the example above, and the parameters may be types, objects, or subprograms. The program unit resulting from instantiation of a generic unit is called an instance of the generic unit -- an instance of a generic subprogram is a subprogram, an instance of a generic package is a package. Generic units are not executable, but instances of generic units are executable.

Ada's generic facility aids greatly in helping the programmer accommodate the strong typing feature of the language, but its usefulness extends much further. The application of generic units to the implementation of reusable software seems quite evident.

NEXT: Ada's Data Types.

Ada is a trademark of the US Department of Defense.

Editor's Note: Seems that ADA has drawn an awful lot of external interest, as well as a great deal of interest from our regular readers.

I get calls from some of you wanting to know what S50 bus machines will run ADA. Well, how about some '**RUMORS**'.

Several I am sure. However, if you go to NCC this summer (July, Chicago) I am betting that GIMIX will have their 68020 running. I predict that it will be running OS-9 - 68K - 68000. Now that should be some sort of **Super S50 machine!** Also I predict that it will be one heck of a fine ADA machine.

It was only back in October of last year that they were denying this but now I am willing to bet my shirt, that is, as concerning a 68020 - GIMIX - OS-9. It will probably be a level one (after all, level one on the 68020, ain't quite the same as level one on a 6809) and things should get better from there. With their intelligent I/O scheme and the 68020 and ????? - who knows what lurks in the skunk-works of GIMIX? Anyway, I like to rumor. Also I like to be right. We will find out this July at NCC. If you don't make it, tune in, for all the latest.

AND, when you do see it - **DON'T TELL ME ANY LONGER THAT THE S50 BUS IS DEAD!! IT IS O.K. - JUST SOME FOLKS SAT ON THEIR RUMPS TOO LONG. HOWEVER, I WILL NOT TRY TO BE A,** "I told you so." But, I did!

DMW

Basic OS-9

Ron Voigt
2024 Baldwin Ct.
Glendale Hts., IL 60137

FINDING FILES

It's been one of my beliefs that if a "little knowledge is dangerous," no knowledge can be catastrophic. My early experiences with computers taught me to learn all that I could. My first job working with a computer was to watch it take data. Occasionally I would answer a question at the terminal to keep it going. Invariably the system would crash between the hours of 1 AM and 4 AM. At this time of night (or morning, depending on your reference) no one wants to be bothered. Calling our resident "computer expert" at some ungodly hour of the morning seldom solved the problem. (Most people don't think to well when they just get up.) The only solution was to learn the system. And I did! Since then my motto has been **LEARN!**

Last month I embarked on a study of the OS-9 disk. My intention was not to turn anyone into a "system's expert". Rather, I hoped you would get a feel for it. OS-9 is integrally tied to its disk. In many systems you can avoid excessive disk interaction. Not in OS-9! Almost anything you do will cause an interaction with the disk. Use a command, the disk drives turn on. Run a program, they grind away. List a file, away they go. You are very dependent on the OS-9 disk.

Just to recap what was talked about last month, we took a look a the first two logical sectors of an OS-9 disk. LSN 0 was the disk's identity sector. Here general information is stored about it. It contains the disk's name, creation date, the disk type, and lots of other goodies. LSN 1 is the disk's sector allocation map. Whether a sector on the disk is in use or not is contained here. Many of the OS-9 commands use these sectors to give you information about the disk. A few of them are FREE, ATTR, and DCHECK.

Another thing from last month was the use of the command DUMP. Entering something like:

```
DUMP /d@
```

will start dumping the entire disk, sector-by-sector. This is fine for the first few sectors, but what if you want to look at LSN 200? DUMP starts to become a little impractical. At the end of this months column is a listing of a Basic09 program called **DiskLook**. DiskLook is made up of a number of procedures. The main procedure is named **DiskLook**. It's subprograms are **Sprint**, **Cprint**, **Get_Sector**, **Zero**, and **Help_List**. The main procedure is menu driven, so all you need do is run it. If you need help just type <H> and it will run Help_List. DiskLook reads sectors from you your disk using the procedure Get_Sector. Sprint prints the information to a standard size screen or the printer. Cprint is for the Color Computer's smaller screen. If you should try to read a non-existent sector, Zero will set all the sector information to zero and an error will be noted. After you run the program, you will see that it is easy to use.

Many of you that have followed my column when it was "**BASIC Basic09**" in the **COLOR MICRO JOURNAL** will remember that I advocated no line numbers. If you look at the main procedure, DiskLook, and at Get_Sector, you're probably saying, "Ah, ha! We caught you! You're using line numbers!" I have to confess, I did use line numbers. There are times when line numbers are justified. In the Get_sector procedure, an ON ERROR GOTO is used which requires line numbers to transfer control to the error trapping routine. In the main procedure I used the ON...GOSUB structure which requires line numbers. I originally used nested IF...THEN...ELSE structures, but they became complex and confusing. After all, the idea is write programs that are

readable. The ON...GOSUB structure makes the program much neater and easier to follow. Also at line 1000 is a subroutine that reads a sector and prints it. I chose to leave this in the main procedure, since it allowed the variables to treated as globals. I didn't have to pass everything back and forth. If it makes you feel better, think of the line numbers as labels.

The output for DiskLook would look like:

```
DiskLook V1.0
LSN:  2      $0002
85/04/09 09:49:59
      0 1 2 3 4 5 6 7 0 2 4 6
ADDR  8 9 A B C D E F 8 A C E
---- ---------------- --------
 00  BF00005504031230 ...U...0
 08  0200000120540C02 .... T..
 10  0000030007000000 ........
 18  0000000000000000 ........
```

This only shows the first 32 bytes of LSN 2. When you use DiskLook, it will actually print addresses $00 thru $FF, but the rest of the listing are all zeros, so I didn't print past byte $1F. LSN 2 is the file descriptor for the root directory on this disk. If you check LSN 0, bytes $08 thru $0A will tell you where this sector is located. Just looking at this would not tell you much. But if we map out the information here, we can get a better idea of what is happening.

ADDR	SIZE	PURPOSE	MY DISK
00	1	Attributes	BF
01	2	Owner	00
03	5	Date last modified	5504031230
08	1	Link count	02
09	4	File size	00000120
0D	3	Date created	540C02
10	F0	Segment list	0000030007

Now the information about the file descriptor for the root directory on this disk is much clearer. Looking at the map, we see that its attributes are $BF. In binary notation this translates to 1 0 1 1 1 1 1 1 which are the attributes d pe pw pr e w r. The most significant bit is set, so this is a file descriptor for a directory. This is where the OS-9 command, ATTR, gets its information. Next is the owner's id, which is $0000. That's the "super user", me!

After this is the date the directory was last modified. Perhaps a file was added or removed. Maybe it was just updated. The 5 bytes here are $5504031230. These are the hex representation of Y M D H M. In decimal they are 85 4 3 14 48 which means the directory was last changed on April 3, 1985 at 2:48 PM (24 Hour Clock). Link count for the directory is 2. Now link count was originally intended to indicate how many directories a file was linked to. As it ended up a file can only be in one directory, so this byte is just there. In my root directories link count is always 2 and it doesn't mean anything. All the other directories and files on the disk are 1. The date the file was created is $540C02. If you guessed this is December 2, 1984, your right!

Finally the segment list contains the locations of the file on the disk. There are 48 5 byte numbers here. Each 5 byte number consist of a 3 byte LSN and a 2 byte size. For the root directory the 5 byte numbers is 0000030007. Which means the directory starts at LSN 3 and is 7 sectors long. After this are all zeros, so there are no more segments in this file. If we want to look at the directory we would start at LSN 3. Here's its beginning:

```
DiskLook V1.0
LSN:  3      $0003
85/04/09 09:52:55
      0 1 2 3 4 5 6 7 0 2 4 6
ADDR  8 9 A B C D E F 8 A C E
---- ---------------- --------
 00  2EAE000000000000 ........
 08  0000000000000000 ........
 10  0000000000000000 ........
 18  0000000000000002 ........
 20  AE00000000000000 ........
 28  0000000000000000 ........
 30  0000000000000000 ........
 38  0000000000000002 ........
 40  4F5339426F6FF400 OS9Boo..
 48  0000000000000000 ........
 50  0000000000000000 ........
 58  000000000000000A ........
 60  66696CE500000000 fil.....
 68  0000000000000000 ........
 70  0000000000000000 ........
 78  000000000000003C .......<
```

This is the actual directory. Each entry is broken into 32 byte segments. The first 29 bytes hold the file's name and the last 3 bytes, its file descriptors location. So the the first entry is from $00 to $1F, the next from $20 to $2F and so forth. Another thing, a file's name has the most significant bit of the last character set. This is the standard way that OS-9 indicates the last character of a string. For example, the last character of "OS9Boo." is $F4 which is really the letter "t" or $74 in Ascii with the 8th bit set.

Another interesting thing about files is the first entry is $2EAE and the second is $AE. Taking into account the setting of the MSB in the last characters, these files are .. and . Sound familiar? If you ever did a **DIR .** or **DIR ..** this is where DIR got the information for the current directory or the one above that one.

The "OS9Boot" file's descriptor is located at LSN $0A. WE can move to this sector.

```
LSN:  10     $000A
85/04/09 09:53:30
      0 1 2 3 4 5 6 7 0 2 4 6
ADDR  8 9 A B C D E F 8 A C E
---- ---------------- --------
0000 030000540C021008 ...T....
0008 0100003032540C02 ...02T..
0010 00000B0031000000 ....1...
0018 0000000000000000 ........
```

I won't go into a complete analysis of this descriptor, but rather point out the highlights. The file was created on $540C02 which is December 2, 1984. That's the same day the directory was created. This file consists of $3032 bytes. It starts at LSN $0B and uses $31 sectors. We can take a look at LSN $0B, where the file begins.

```
DiskLook V1.0
LSN:  11     $000B
85/04/09 09:55:14
      0 1 2 3 4 5 6 7 0 2 4 6
ADDR  8 9 A B C D E F 8 A C E
---- ---------------- --------
0000 87CD030B000EE181 ........
0008 D3001500ADFF4343 ......CC
0010 446973EB0216000F Dis.....
0018 1600511600CC1600 ..Q.....
0020 491602771600434F I..w..CO
0028 B7006F8EFF403008 ..o..@0.
0030 86D0A78417025DA6 ......].
```

If you look at $0E thru $12, you'll see the name "CCDIS.". (Remember the 8th bit being set?) The $EB at $13 is really $6B. That's the ascii letter "k". So this is CCDISK. It is the first module of my "Boot" file. If we were to look at the remainder of this file we would see items like BOOT, IOMAN, /D0, SCFMAN, and so on. These modules make up OS9B00T.

Files can come in many types and sizes. The OS9B00T is a special one. Some are executable modules like DIR, ATTR, and COPY. Others are ascii text files. That's how I store records of this column, papers for school and correspondence. Even the directory is a file of file names. They all have a file descriptor and are allocated space on the disk.

Stop and study your disks with DiskLook. You can trace directory paths and locate physical storage on the disk. I think you'll be amazed at how well they are laid out and how well everything works. You'll start to appreciate what happens every time you use a Disk I/O. Every time you access the disk many things are happening. Directories are searched, file descriptors found and disk sectors accessed. And all this happens in seconds. Play around with Disklook and do some of your own investigating.

Take care until next next time!

LISTING FOR DiskLook

```
PROCEDURE DiskLook
(* This program will examine and print
(* the contents of a sector on an OS-9 disk.
(*
DIM sector(256):BYTE
DIM dr,vector,path,lsn:INTEGER
DIM ans,choice:STRING[1]; path_name:STRING[64]
DIM fflag:BOOLEAN
(*
(* We set up the print parameters
PRINT CHR$(12)
PRINT
PRINT "Disk Look    Version 1.00"
PRINT "by Ron Voigts"
PRINT "for The '68' Micro Journal"
PRINT
PRINT "<L>ong or <S>hort listing? ";
GET #1,ans
IF ans="L" OR ans="l" THEN
fflag:=TRUE
ELSE
fflag:=FALSE
ENDIF
PRINT
PRINT
PRINT "Which disk drive? ";
GET #1,ans
dr:=VAL(ans)
path_name:="/d"+STR$(dr)+"@"
PRINT
PRINT
(*
(* This loop handles I/O with the operator
LOOP
PRINT
PRINT ">>";
GET #1,ans
PRINT
RESTORE
REPEAT
READ choice,vector
DATA "F",1,"f",1,"B",2,"b",2,"S",3,"s",3
DATA "P",4,"p",4,"R",5,"r",5,"H",6,"h",6
DATA "Q",7,"q",7
DATA "?",-1
EXITIF ans=choice THEN
ON vector GOSUB 100,110,120,130,140,150,160
ENDEXIT
UNTIL vector=-1
IF vector=-1 THEN
PRINT
PRINT "Type <H> for help."
PRINT
ENDIF
ENDLOOP
(*
(* move a sector forward
100 lsn:=lsn+1
GOSUB 1000
RETURN
(*
(* move a sector back
```

```
110 IF lsn>0 THEN
lsn:=lsn-1
GOSUB 1000
ENDIF
RETURN
(*
(* select a new sector
120 INPUT "LSN>>",lsn
GOSUB 1000
RETURN
(*
(* print sector to printer
130 OPEN #path,"/P":WRITE
PRINT #path
PRINT #path
RUN Sprint(lsn,path,sector)
CLOSE #path
RETURN
(*
(* redisplay sector
140 IF fflag=TRUE THEN
RUN Sprint(lsn,1,sector)
ELSE
RUN Cprint(lsn,sector)
ENDIF
RETURN
(*
(* print help list
150 RUN help_list
RETURN
(*
(* return to the system
160 PRINT
END "  --Disk Look Terminated--"
(* do the read and print sector
1000 RUN get_sector(path_name,lsn,sector)
IF fflag=TRUE THEN
RUN Sprint(lsn,1,sector)
ELSE
RUN Cprint(lsn,sector)
ENDIF
RETURN

PROCEDURE zero
(* This module zeroes a sector *)
(*
PARAM sector(256):BYTE
DIM i:INTEGER
FOR i=1 TO 256
sector(i):=0
NEXT i
END

PROCEDURE Sprint
(* This is the standard print routine
(* It's output is 64 characters wide and
(* it can output to any path
(* pointed to by 'p'.
(* Disk Look uses the standard output
(* or a path to the printer
(*
PARAM l,p:INTEGER; s(256):BYTE
DIM i,j:INTEGER
PRINT #p,"Disk Look  V1.0       ";
PRINT #p USING "s5,16,x1,s,h4","LSN: ",l,"S",l;
PRINT #p,"       "; DATE$
PRINT #p
PRINT #p,"ADDR   0 1  2 3  4 5  6 7  8 9  A B";
PRINT #p,"  C D  E F  0 2 4 6 8 A C E"
PRINT #p,"----  ---- ---- ---- ---- ---- ----";
PRINT #p,"  ---- ----  ----------------"
FOR i=1 TO 16
PRINT #p USING "x1,h2,x1,s1",i*16-16," ";
FOR j=1 TO 16
IF MOD(j,2)=1 THEN
PRINT #p," ";
ENDIF
PRINT #p USING "H2",s(i*16-16+j);
NEXT j
PRINT #p,"  ";
FOR j=1 TO 16
IF s(i*16-16+j)<=$7F AND s(i*16-16+j)>=$20 THEN
PRINT #p,CHR$(s(i*16-16+j));
ELSE
PRINT #p,".";
ENDIF
NEXT j
PRINT #p
NEXT i
END

PROCEDURE get_sector
(* here we get a sector from the disk
(* that is named in 'name' at lsn
(*
PARAM name:STRING[64]
PARAM lsn:INTEGER; sector(256):BYTE
DIM path:BYTE; errnum:INTEGER; location:REAL
(*
(* set up for possible disk errors *)
ON ERROR GOTO 1000
(*
(* Read a sector *)
location:=256.*FLOAT(lsn)
OPEN #path,name:READ
SEEK #path,location
GET #path,sector
100 CLOSE #path
END
(*
(* This area handles reading to far
1000 errnum:=ERR
PRINT "ERR #"; errnum
IF errnum=211 THEN
PRINT " --Beyond End Of Disk--"
ENDIF
RUN zero(sector)
PRINT
GOTO 100

PROCEDURE help_list
(* This module prints a list of commands
(*
PRINT CHR$(12)
PRINT
PRINT "    Help   List"
PRINT "  ****************"
PRINT "F - move a sector forward"
PRINT "B - move a sector backward"
PRINT "$ - enter another sector"
PRINT "P - display sector to printer"
PRINT "R - redisplay the sector"
PRINT "H - get help list"
PRINT "Q - quit Disk Look"
PRINT
END

PROCEDURE Cprint
(* This procedure prints a shortened
(* version of the screen dump for
(* for the Color Computer's screen.
(*
PARAM l:INTEGER; s(256):BYTE
DIM i,j:INTEGER
PRINT "Disk Look  V1.0"
PRINT USING "s5,16,x2,s,h4","LSN: ",l,"S",l
PRINT DATE$
PRINT
PRINT "      0 1 2 3 4 5 6 7 8 9 A B C D E F "
PRINT "ADDR                  --------"
PRINT "---- ---------------- --------"
FOR i=1 TO 32
PRINT USING "x1,h2,x1,s",i*8-8," ";
FOR j=1 TO 8
PRINT USING "h2",s(i*8-8+j);
NEXT j
PRINT " ";
FOR j=1 TO 8
IF s(i*8-8+j)<=$7F AND s(i*8-8+j)>=$20 THEN
PRINT CHR$(s(i*8-8+j));
ELSE
PRINT ".";
ENDIF
NEXT j
PRINT
NEXT i
END
```

CoCo User Notes

by Carl Mann
30 Warren Ave.
Amesbury, MA 01913

MODEM COMMUNICATIONS, Part Two
--- or ---
...And a BBS in the Bedroom

Ahh, the joys of bit-banging! It's sooo easy to do it with CoCo: the 4-pin port marked "RS-232" on the rear apron of the cabinet (okay, the box!) will drive a printer, a modem, or even (as I do at my daytime job) anywhere from 100 to 200 feet of cable between CoCo and a "port" (a place where data bits are stuffed into or out of a computer) on a much larger machine. (Not necessarily "better", mind you. The particular dinosaur of which I write is at least two software generations behind CoCo in terms of versatility, variety, and what I call the "**Victory Factor**": the chance that the owner will succeed in fixing whatever might go wrong (or recover from it without damage) without needing a master's degree in computer science and the like to do it.)

Ah, but I digress. CoCo's no dinosaur, and that's who we're exploring today. (Gee, the world is such a different place when seen from atop a soapbox!) (Thirty years ago, you could still jump up and down on one. Now they turn to a mass of soggy pasteboard when it rains...)

The business is modems and BBs's. If you have a certain adventurous panache in your outlook, you might like the idea of setting up shop yourself as a CoCo sysop. (Does THIS kind of life look interesting to YOU? Night after night, dinner with the printer! Crackers, crumbs, logic bombs, weekends performing mailing list and software maintenance, plus ten thousand laminated, embossed business cards neatly imprinted in gold with your logo, full name, and the wrong address...) And yet, for all that, the people I know who run their own computer Bulletin Board Services (longhand for "BBS"s - the people are "System Operators", or "sysops" for short) practice some of the finest qualities of which the human race is capable. It's a tough gig, and (unless you've the heart of a genuine businessman) can be pretty costly. If you're not warned off by now, though, you ain't a-gonna be. Let's see what you need to do the job.

First: the software. (It's a principle. Software FIRST. Always. Think about it.) There are several commercial sources for CoCo BBS packages. Most of them cost about a hundred bucks or so. Check the pages of Hot CoCo, CoCoAds, or the Rainbow for what's available now. It's a growing family.

If a hundred bucks is too much, (it is for me) there is another way. Pick up the 'phone and call up Falsoft in Louisville, Kentucky. Order the "Rainboard" software and the articles that appeared as documentation. This package is available for less than $35.00, including the cost of the Star-Kits "Remoterm" terminal driver package that composes the heart of the system. (Star-Kits advertises in 68' Micro **Journal.**)

As for hardware: The Rainboard requires at least one disk drive. (If you have two, that's better. If they're double-sided, so much the better. In any case, try for a moment to imagine either a RAM- or tape-based, diskless BBS. Write me when you get the bugs worked out.) You also need an auto-answer, direct connect modem of some kind. (Po' boys an' gals, you might start your search at Computer Products and Peripherals, Unlimited in Haverhill, Massachussetts if nobody in your neighborhood happens to have a good unit in the closet. Also try Metric Resources, Inc. at 1-800-368-2764. Both companies have mailing lists. Get on 'em.)

Oh, yes - you'll need a 'phone line. (Unless you like the idea of friends and neighbors calling up for whatever reason, only to be greeted by the ear-splitting whistle of an automatic modem in perpetual search of a convenient mate, of course.) As long as you ask for no money over the new connection, Ma Bell will be merciful and only charge you the residential rate. Ask for even a ten-cent subscription fee, though, and she'll show you what makes the business world go 'round REAL quick. The moral: If you think you're going to run a business through a BBS, analyze the costs VERY carefully. Then set your prices accordingly.

Got specific questions about modems, software, and the like? Drop me a note; I'll do you right. (Send a SASE for an up-to-date nationwide list of active BBS's of all kinds.) And keep your eyes open for something called "Packet Radio". It's new, it takes a ham license, and it eliminates the cost of 'phone time. (Get a-holt of a Heathkit catalog for starters. I'll write a bit about it later. It's new to me, too.)

Until next time ...

Using FLEX/Star-DOS

by Troy Brumley
8552 Huddleston Drive
Cincinnati, OH 45236

HOW TO "INSTALL" FLEX
or
MAKING "SYSTEM" DISKS

Two month's ago we briefly discussed FLEX. This month we will install FLEX on a 64K CoCo. For convenience I will discuss the F-MATE FLEX Version 3.0 sold by DATACOMP since it is the version that I have. From talking with Bob Nay and reading Ron Anderson's FLEX USERS NOTES Columns I get the impression that STARDOS really is equivalent to FLEX, so the CoCo version should be nearly identical to the CoCo FLEX version I'll describe here. I imagine that other CoCo FLEX Systems should work in a similar manner.

BY HIS BOOTSTRAPS

"Install" is an intimidating word. Actually, Computer no modifications are required unless you don't have 64K of RAM yet, which is required to be able to run FLEX on a CoCo. The process we will describe consists essentially of making one or more (probably MORE) copies of the delivery disk.

The delivery disk is a FLEX BOOT DISK. This means it contains a short program that will load FLEX into your CoCo and give FLEX control of the system. This process is known as "BOOTing the System".

BUT FIRST...

****
* WRITE PROTECT YOUR DISK! *
****

This applies to any software you buy, whether it is for FLEX, OS-9, EXTMD9 or CoCo DOS. Always cover the notch on the right side of your disk with a WRITE PROTECT TAB (that's a fancy name for an opaque piece of tape; so if you don't have some Write Protect Tabs laying around, use a piece of tape that will NOT let light through).

You should never need to update a delivery disk. I personally think that all software should come on write protected disks. This avoids any chance of accidental update or erasure of a disk.

Enough of my opinions...

WHAT SHOULD YOU HAVE

The package I got from DATACOMP contained 1 delivery disk, a small manual containing the TSC documentation on FLEX and their Assembler and Editor, and a special Conversion Manual from DATACOMP.

Resist the urge to boot up FLEX and play. Read thru the Conversion Manual. In it you will find helpful hints and lucid instructions. Bob Nay worked hard on that manual, and I'd hate to see any of that effort go to waste.

Take good care of the disk. DATACOMP requires that you return that disk to them for any upgrades or enhancements. This saves them the trouble of keeping track of serial numbers, owner changes, and such; whoever has the Master Disk gets an update. No disk, no updates. If you have FHL FLEX or XEX, I seem to recall hearing that your FLEX boot disk is copy protected. If that is true, you will have only the disks that you bought to boot with. You should be EXTREMELY careful with it (or them).

NOW TO BOOT

Take your disk (you did write protect it, didn't you?) and insert it into your drive 0. If you have only 1 drive, that's drive 0. If you have 2 or more drives, drive 0 is the drive that the system uses as a default when you load a program.

Now type 'RUN "FLEX"' and hit <ENTER>.

The disk will spin and messages will appear informing you that the FLEX BOOT PROGRAM is being loaded and executed. After a few seconds the disk will stop spinning and FLEX will ask you to enter the current date. I am writing this article on March 10th of 1985, so I entered "03 10 85". (P.S. You don't need to enter leading zeros here.)

After you enter the date your disk will spin some more and FLEX will finish boot-

Continued on Page 35

FINALLY !!

Now you can run

TSC XBASIC Programs Compiled to Asmb. Lang., under **OS-9™, CoCo OS-9,or FLEX™ with**

☆ K-BASIC

K-BASIC under **OS-9** and **FLEX** will now compile **TSC BASIC, XBASIC, and XPC Source Code Files**

K-BASIC now makes the multitude of **TSC XBASIC** Software available for use under **OS-9**. Transfer your favorite **BASIC** Programs to **OS-9**, compile them, Assemble them, and **BINGO** -- usable, multi-precision, familiar Software is running under your favorite Operating System!

K-BASIC (**OS-9** or **FLEX**), **including** the **Assembler**

$199.00

TELEX 558 414 PVT BTH

(615)842-4600

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

SCULPTOR

Microprocessor Developments Ltd.'s Commercial Application Generator Program provides a **FAST** Commercial Application Development tool unavailable to the OS-9 and UniFLEX User before. Develop any Commercial Application in 20% of the normal required time; gain easy updating or customizing. **PLUS**, the Application can also be run on MS-DOS and Unix machines! **Sculptor** handles input validation, complex calculations, and exception conditions as well as the normal collecting, displaying, reporting, and updating information in an orderly fashion. Key fields to 160 bytes; unlimited record size; file size should be held to 17 million records. Utilizes **ISAM** File Structure and B-tree Key files for rapid access. Input and Output communication with other programs and files plus a library of ISAM routines for use with C Programs. Run-time included w/ the Development package; a compiled Application only needs a Run-time License. Additional charge for Networked Units. Prices for **Development Package/Run-time**. Discounts available for purchases of 5 or more Run-time Packages.

UniFLEX -- $1450.00/337.00
OS-9 Lvl II -- $995.00/200.00

Unix -- up to $2535/675 -- Call for info!
IBM PC & compats -- $635.00/170.00

☆ **Basic09 Tour Guide**

by Dale Puckett -- An excellent Book on using OS-9. Oriented towards using the powerful **Basic09 Programming Language**, it also contains a lot of good information on using OS-9 in general.
Normally $18.95
Special ---- NOW only $16.00

☆ **---- FLEX Software ----**

TSC "Flex Utilities"	was $75.00,	**NOW only $65.00**
TSC "Sort Merge"	was $75.00,	**NOW only $65.00**
TSC "6809 Basic"	was $75.00,	**NOW only $65.00**
TSC "Extended Basic"	was $100.00,	**NOW only $90.00**
TSC "DeBug"	was $75.00,	**NOW only $65.00**
TSC "FLEX Diagnostics"	was $75.00,	**NOW only $65.00**
TSC "Text Processing System"	was $75.00,	**NOW only $65.00**
TSC "68000 Cross Assembler"	was $250.00,	**NOW only $199.95**

VISA

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9 FLEX SOFTWARE

Availability Legends ---
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

!!! Please Specify Your Operating System & Disk Size !!!

☎ TELEX 558 414 PVT BTH

(615)842-4600

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

PROGRAMMING

ASSEMBLERS

ASTRUK09 from **Southeast Media** -- A "Structured Assembler for the 6809" which requires the **TSC Macro Assembler.** F, CCF - $99.95

Macro Assembler for TSC -- The FLEX STANDARD Assembler.
Special -- CCF $35.00; F $50.00

OSM Extended 6809 Macro Assembler from **Lloyd I/O.** -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. Generate OS-9 Memory modules under FLEX. FLEX, CCF, OS-9 $99.00

Relocating Assembler w/Linking Loader from **TSC.** -- Use with many of the C and Pascal Compilers. F,CCF $150.00

MACE, by Graham Trott from **Windrush Micro Systems** -- Co-Resident Editor and Assembler; fast interactive A.L. Programming for small to medium-sized Programs. F,CCF - $75.00

XMACE -- MACE w/ Cross Assembler for 6800/1/2/3/8 F,CCF - $98.00

TRUE CROSS ASSEMBLERS from **Computer Systems Consultants** -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/146805, 6809/00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/40/48/C48/49/C49/50/8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.
FLEX, CCF, OS-9, UniFLEX each - $50.00
any 3 - $100.00
the complete set w/ C Source (except the 68000 Source) - $200.00

XASM Cross Assemblers for FLEX from **Compusense Ltd.** -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Soruce Code format, Assembler options, etc., in providing code for the target CPU's. Complete set, FLEX only - $150.00

CRASM8 from **Lloyd I/O** -- 8-Bit Macro Cross Assembler with same features as OSM; cross-assemble to 6800/1/2/3/4/5/8/9/11, 6502, 1802, 8048 Sers, 80/85, Z-8, Z-80, TMS-7000 sers. Supports the target chip's standard mnemonics and addressing modes.
FLEX, CCF, OS-9 Full package -- $399.00

CRASM 16.32 from **Lloyd I/O** -- Cross Assembler for the **68000.**
FLEX, CCF, OS-9 $249.00

DISASSEMBLERS

SUPER SLEUTH from **Computer Systems Consultants** -- Interactive Disassembler; extremely POWERFUL! Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems

Color Computer	SS-50 Bus (all w/ A.L. Source)
CCO (32K Req'd) Obj. Only $49.00	F, $99.00
CCF, Obj. Only $50.00	U, $100.00
CCF, w/Source $99.00	O, $101.00
CCO, Obj. Only $50.00.	

DYNAMITE + from **Computer Systems Center** -- Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.

CCF, Obj. Only $100.00	CCO, Obj. Only $ 59.95
F, " " $100.00	O, " " $150.00
	U, " " $300.00

PROGRAMMING LANGUAGES

PL/9 from **Windrush Micro Systems** — By Graham Trott. A combination Editor/Compiler/Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ Page Manual with tutorial guide. F, CCF - $198.00

WHIMSICAL from **Whimsical Developments** -- Now supports **Real Numbers.** "Structured Programming" WIT OUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, **6-digit Real Numbers,** unlimited sized Arrays (vectors only); interrupt handling; long variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted as called during compilation. Normally produces 10% less code than PL/9. F and CCF - $195.00

C Compiler from **Windrush Micro Systems** by James McCosh. Full C for FLEX except bit-fields, including an Assembler. Requires the TSC Relocating Assembler if user desires to implement his own Libraries. F and CCF - $295.00

C Compiler from **Introl** -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.
F, CCF, and O - $375.00 U - $425.00

PASCAL Compiler from **Lucidata** -- ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.
F and CCF 5" - $190.00 F 8" - $205.00

PASCAL Compiler from **OmegaSoft** (now **Certified Software**) -- For the **PROFESSIONAL;** ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Powerful; Flexible. Requires a "Motorola Compatible" Relo. Asmb. and Linking Loader.
F and CCF - $425.00 One Year Maint. - $100.00

K-BASIC from **LLOYD I/O** -- A "Native Code" **BASIC Compiler** which is now **Fully TSC XBASIC compatible.** The compiler compiles to Assembly Language Source Code. A NEW, **streamlined,** Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.
FLEX, CCF, OS-9 Compiler **with** Assembler - **$199.00**

CHURCH COBOL from **Compusense Ltd.** -- Supports large subset of ANSII Level 1 COBOL with many of the useful Level 2 features. Full FLEX File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System.
FLEX, CCF; Normally $199.00
Special Introductory Price (while in effect) -- **$99.95**

FORTH from **Stearns Electronics** -- A CoCo **FORTH** Programming Language. **Tailored to the CoCo!** Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!
Color Computer **ONLY** - $58.95

VISA MasterCard

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

SOUTH EAST MEDIA
5900 Cassandra Smith Rd. CoCo OS-9™ FLEX™
Hixson, TN 37343 SOFTWARE
Info (615) 842-4601

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware

GENERAL

SOFTWARE DEVELOPMENT

BasicO9 IRef from Southeast Media -- This **BasicO9 Cross Reference** Utility is a BasicO9 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a **Program List Utility** which outputs a fast "pretty printed" listing with line numbers. Requires BasicO9 or RunB.
O & CCO obj. only -- $39.95; w/ Source - $79.95

Lucidata PASCAL UTILITIES (Requires LUCIDATA Pascal ver 3)

IREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary; unlimited nesting capabilities.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.

F, CCF --- **EACH Utility** 5" - $40.00, 8" - $50.00

DUB from Southeast Media -- A **UniFLEX** "basic" De-Compiler. Re-Create a Source Listing from UniFLEX **Compiled basic** Programs. Works w/ ALL Versions of 6809 UniFLEX basic. U - $219.95

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants -- TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
F and CCF, U - $25.00, w/ Source - $50.00

DISK UTILITIES

OS-9 VDisk from Southeast Media -- For **Level I** only. Use the **Extended Memory** capability of your SWTPC or Gimix CPU card (or similar format RAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
-- **Level I ONLY** -- OS-9 obj. only - $79.95; w/ Source - $149.95

O-F from Southeast Media -- Written in **BASICO9 (with Source)**, includes: **REFORMAT**, a BASICO9 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX; and **FLEX**, a BASICO9 Program that does the actual **read** or **write** function to the special **O-F Transfer Disk**; user-friendly menu driven. Read the FLEX Directory, Delete FLEX Files, Copy both directions, etc. FLEX users use the special disk just like any other FLEX disk. O - $79.95

COPYMULT from Southeast Media -- Copy **LARGE** Disks to several **smaller disks.** FLEX utilities allow the backup of **ANY size disk** to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by **COPYMULT**. No fooling with directory deletions, etc. **COPYMULT.CMD** understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes **BACKUP.CMD** to download any size "random" type file; **RESTORE.CMD** to restructure copied "random" files for copying, or recopying back to the host system; and **FREELINK.CMD** as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.
Completely documented Assembly Language Source files included.
ALL 4 Programs (FLEX, 8" or 5") $99.50

COPYCAT from Lucidata -- Pascal NOT required. Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in **Modular Source Code** (Assembly Language) to help solve unusual problems.
F and CCF 5" - $50.00 F 8" - $65.00

(615)842-4600
TELEX 558 414 PVT BTH
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

FLEX DISK UTILITIES from Computer Systems Consultants -- Eight (8) different Assembly Language (w/ Source Code) FLEX Utilities for every FLEX Users Toolbox: **Copy** a file with **CRC Errors**; **Test** Disk for errors; **Compare** two Disks; a fast **Disk Backup** Program; **Edit Disk Sectors**; **Linearize Free-Chain** on the Disk; print **Disk Identification**; and **Sort** and **Replace** the Disk Directory (in sorted order). -- **PLUS** -- Ten XBASIC Programs including: A **BASIC Resequencer** with **EXTRAS** over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs **Compare, Merge,** or **Generate Updates** between two BASIC Programs, check BASIC Sequence Numbers, compare two **unsequenced** files, and 5 Programs for establishing a **Master Directory** of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in **TSC BASIC, XBASIC,** and **PREDECESSOR** BASIC Programs.
ALL Utilities **Include Source** (either BASIC or A.L. Source Code).
F and CCF - $50.00
BASIC Utilities ONLY for **UniFLEX** -- $30.00

COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".
FLEX, CCF, OS-9, UniFLEX; with complete Source - $100.00
without Source - $90.00

XDATA from Southeast Media -- A **COMMUNICATION** Package for the **UniFLEX** Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using Checksum or CRC; Re-Transmits bad blocks, etc. U - $299.99

GAME

RAPIER - 6809 Chess Program from Southeast Media -- Requires FLEX and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. This is one of the **strongest** CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels). F and CCF - $79.95

VISA

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

SOUTH EAST MEDIA
5900 Cassandra Smith Rd. Hixson, TN 37343
CoCo OS-9™ FLEX™
SOFTWARE
info (615) 842-4601

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

TELEX 558 414 PVT BTM
(615)842-4600
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

BUSINESS

WORD PROCESSING

SCREDITOR III from **Windrush Micro Systems** -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store comman command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX or SSB DOS, OS-9 - **$175.00**

STYLO-GRAPH from **Great Plains Computer Co.** -- A full-screen oriented WORD PROCESSOR — (uses the 51 x 24 Display Screens on CoCo FLEX/STAR-DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.
NEW PRICES —> CCF and CCO - $99.95, F or O - $179.95, U - $299.95

STYLO-SPELL from **Great Plains Computer Co.** -- Fast Computer Dictionary, Complements Stylograph.
NEW PRICES —> CCF and CCO - $69.95, F or O - $99.95, U - $149.95

STYLO-MERGE from **Great Plains Computer Co.** -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.
NEW PRICES —> CCF and CCO - $59.95, F or O - $79.95, U - $129.95

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!
F or O - **$329.95**, U - **$549.95**

JUST from **Southeast Media** -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with ANY Editor. Supplied with "Structured Source" (Windrush PI/9); easy to see the flow of the program.

F and CCF - **$49.95**

SPELL8 "Computer Dictionary" from **Southeast Media** -- **OVER 120,000 words! Look up a word** from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX (tCS). Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELL8 first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELL8 also allows the use of **Small Disk Storage** systems.

F and CCF - **$129.95**

DATA BASE - ACCOUNTING

XDMS from **Westchester Applied Business Systems** -- Powerful DBMS; N.L. program will work on a single sided 5" disk, yet is F-A-S-T. Supports Relational, Sequential, Hierarchical, and Random Access File Structures; has Virtual Memory capabilities for Giant Data Bases. **XDMS Level I** provides an "entry level" System for defining a Data Base, entering and changing the Data, and producing Reports. **XDMS Level II** adds the POWERFUL "GENERATE" facility with an English Language Command Structure for manipulating the Data to create new File Structures, Sort, Select, Calculate, etc. **XDMS Level III** adds special "Utilities" which provide additional ease in setting up a Data Base, such as copying old data into new Data Structures, changing System Parameters, etc.

XDMS System **Manual** - **$24.95**
XDMS Lvl I - F & CCF - **$129.95**
XDMS Lvl II - F & CCF - **$199.95**
XDMS Lvl III - F & CCF - **$269.95**

ACCOUNTING PACKAGES -- **Great Plains Computer Co.** and **Universal Data Research, Inc.** both have Data Base and Business Packages written in TSC xBASIC for FLEX, CoCo FLEX, and UniFLEX.

MISCELLANEOUS

TABULA RASA SPREADSHEET from **Computer Systems Consultants** -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires **TSC's Extended BASIC.**

F and CCF, U - **$50.00**, w/ Source - **$100.00**

DYNACALC from **Computer Systems Center** -- Electronic Spread Sheet for the 6809.

F and **SPECIAL** CCF - **$200.00**, U - **$395.00**

FULL SCREEN INVENTORY/MRP from **Computer Systems Consultants** -- Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. **Requires TSC's Extended BASIC.**

F and CCF, U - **$50.00**, w/ Source - **$100.00**

FULL SCREEN MAILING LIST from **Computer Systems Consultants** -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. **Requires TSC's Extended BASIC.**

F and CCF, U - **$50.00**, w/ Source - **$100.00**

DIET-TRAC Forecaster from **Southeast Media** -- An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P GS) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.

F - **$59.95**, U - **$89.95**

VISA
** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign
SOUTH EAST MEDIA
5900 Cassandra Smith Rd
Hixson, TN 37343
into (615) 842-4601
Sale ONLY FOR SOFTWARE

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

ing. The most obvious sign that FLEX is up is that your screen will be in a 51 column by 24 line display format. If you have a poor quality TV you could have some trouble reading this screen at first. Try turning down the color on your TV. If this doesn't help you may want to buy a monitor or a better TV.

If you just can't read the screen comfortably you can switch into 32 character mode by typing 'V32' and hitting <ENTER>. This will return your display to the mode you normally use in BASIC. If you want to switch back to the hi-res screen format type 'V51' and hitting <ENTER>.

FLEX IS WAITING ON YOU

Up near the upper left hand corner of the screen you will see three plus signs and a black block that blinks on and off. The three plus signs are FLEX's PROMPT, much like BASIC's OK. The black box is the cursor. FLEX is ready and willing to do your bidding.

OK, IT'S TIME TO PLAY

Let's enter a command. The FLEX command to list the names of all the files on a disk is CAT. Type in 'CAT' and hit <ENTER>.

Your disk will start to spin and then a listing of the files on the disk print on the screen. When the screen is full FLEX will pause and wait for you to hit the <ESCAPE> key before the list continues.

In DATACOMP's FLEX the <ESCAPE> key is defined as the <1> key (because that is where the "Escape" Key in on normal Terminals). Hit it and another full screen of files will list out. When FLEX pauses again just hit <ENTER>. There are a lot of files on the delivery disk and we don't need to see them all now.

HOW TO INSTALL

Before we play some more we really should create a couple of bootable System Disks so we can take our master disk and hide it in a safe place. The conversion manual that comes with the DATACOMP FLEX system has an excellent explanation of how to create a new boot disk, so I won't go into quite the same level of detail that it does. I'll say just enough so you can get the idea of how to do it.

DOUBLE DRIVE SYSTEMS

Take the FLEX disk out of drive 0 and double check it to see that it is write protected. Once it is, put it back in drive 0.

Put a blank disk in drive 1 and enter 'NEWDISK 1' (some other Systems may use "FORMAT" instead of "NEWDISK") at the +++ prompt. Drive 0 will spin and then your CoCo will start asking questions about the disk you want to format. To keep things simple let's use the same disk format that the FLEX disk is in, single sided, double density, 35 tracks.

Once you have entered all the necessary information drive 1 will start to spin and the disk will be formatted. The complete format and verify operation takes around a minute on my system; single density takes less time, while double sided takes longer.

Once formatting is complete enter 'MAKESYS 1' at the +++ prompt. Drive 0 will spin and then drive 1 will start to spin. In less that a minute the +++ prompt will reappear. FLEX has taken a small piece of the disk away from itself and set it up so that it has the FLEX/BAS and a Machine Language file that must be run to boot FLEX from CoCo DOS, and therefore must be in RS Format.

Now enter 'COPY 0 1' at the +++ prompt. Go get a cup of coffee or a coke. This takes around 5 minutes on my system. FLEX will copy every file from the disk in drive 0 to drive 1.

When the +++ prompt returns you will have a good copy of the delivery disk in drive 1. Take that delivery disk and hide it in a safe place; NOW!

SINGLE DRIVE SYSTEMS

If you have only 1 drive this will take a while. The installation procedure is similar to the one that I described for a double drive system, but you will have to swap disks frequently to complete the process. While this is a very tedious procedure I know it can be done. I did it when I bought FLEX. I acquired my dual drive system only two months ago and I've had FLEX for almost nine months (and is FLEX or STARDOS nice with two Drives).

To format a blank disk we will still use the NEWDISK command, but instead of

entering 'NEWDISK 1' enter 'NEWDISK 0'. FLEX will ask if you have a scratch disk in drive 0 before he formats the disk, so you have plenty of time to swap disks.

After the disk is formatted, instead of entering 'MAKESYS 1', put your FLEX delivery disk back in your drive and enter 'MAKESYS 0'. As with the single drive NEWDISK operation, FLEX will give you an opportunity to remove the system disk and insert the new disk.

The COPY command is specifically for multi-drive systems. In order to COPY files on a single drive system, the SDC (Single Drive Copy) command should be used. Unfortunately, you have to enter each individual file name to copy, and SDC will only copy 5 files at a time. To make your job easier you will want to get a copy of the disk CATALOG to use as a checklist.

If you have a printer connected to the SERIAL I/O port on the back of your CoCo you can get a catalog listing with no trouble, as long as the printer is running at 600 baud. If you have one of the many non-Tandy printers and a special serial to parallel converter box, set it to 600 baud. I'll show you how to change FLEX's baud rate next month. If you have a non-Tandy serial printer (I have an OKIDATA Microline 82A running at 1200 baud) make sure it is set up to run at 600 baud also. If you have any of the normal Tandy printers, you're all set.

Enter 'P CAT' at the +++ prompt. A complete list of all files on you disk will be printed for you. You can use this as a check list for you single drive copy. Copy 5 files at a time, and be sure not to get your hands crossed as you swap disks. Once all the files have been copied, hide the original master in a safe place (now, doesn't "COPY 0 1" look a lot easier?).

WHAT'S NEXT

I've had a busy couple of months so I didn't get to cover everything I wanted to. Next month I'll have a short section on customizing the printer drivers for non-standard printers, and I'll start explaining some of the powerful features of FLEX, such as I/O re-direction and the EXEC processor.

If you have any questions on FLEX and the CoCo feel free to drop me a line care of '68', but be sure to include a stamped and self addressed envelope.

Until next time...

Implementing TYPEAHEAD in FLEX

E. M. (Bud) Pass, Ph.D.
Computer Systems Consultants
1454 Latta Lane, N. W.
Conyers, GA 30207
404-483-1717/4570

INTRODUCTION

Several months ago, an article was published in '68' Micro Journal which provided a means of implementing type-ahead in FLEX. Unfortunately, it was not compatible with spooling nor with any programs such as full-screen editors which use a large number of control characters; it also did not handle the full buffer situation.

This article provides an alternate means of implementing typeahead in FLEX which is compatible with spooling and full-screen editors and also occupies substantially less memory than the original routine.

DISCUSSION

The theory behind the implementation of typeahead in FLEX is reasonably simple, given a device such as an ACIA or PIA which is capable of generating an interrupt when it receives an input character and has at least a one-byte buffer to hold the character until it can be processed.

When a character is received, it is placed into the circular queue by the interrupt routine, assuming that the queue is not full. When a program (or FLEX itself) requires input, the character is taken from the queue, unless it is empty, in which case the requestor is caused to wait until the queue is not empty. When a character is taken from the buffer, it is optionally echoed, so the results on the terminal are the same as if typeahead had not been active. This is the same technique used by OS/9, but UNIX and UNIFLEX echo the characters as they are entered.

Since every character input is potentially critical to a program such as a full-screen editor, the typeahead routine must not use or intercept any of the input characters. This is the reason the original routine was not compatible with full-screen editors, as it used several of the control codes for special purposes.

Without the output pause option of the original typeahead routine and without the ability to pause a listing by striking a key (it is lost because of the ambiguity of whether a character is for input or pause purposes), there are still two manners in which to pause output. One is to set the FLEX TTYSET DP page depth parameter to cause output to pause at the end of each screen page. The other (non-exclusive) means is to attach a switch to the CTS line on the ACIA or other interface device to inhibit transmission temporarily. There are several peculiarities in FLEX's handling of the DP parameter, primarily related to pausing the printer as if it were the terminal, but they are avoidable.

In order to receive control properly when an interrupt occurs, the typeahead routine must insert its entry into the interrupt queue. It must also return control to the original entry at the head of the interrupt queue. This is the reason that the original routine was not compatible with spooling, which is the only interrupt-driven routine supported by FLEX.

TECHNIQUES

The construction of a circular queue is a fairly simple programming task. It may be accomplished in one of several manners. The one used in this case involves the use of two pointers and one counter. It could have been accomplished with the use of two pointers without the counter. One pointer indicates the location into which the next arriving character will be placed. The other pointer indicates the location from which the next requested character will be taken. When either pointer reaches the end address of the buffer area, it is reset to the beginning of the area; otherwise, pointers values increase by one character for each insertion or deletion. The counter indicates the number of characters in the queue. When the queue is full, no additional characters are placed into it. When the queue is empty, the terminal status is "not ready", otherwise it is "ready".

The typeahead routine replaces the FLEX input vectors at $CD0A, $CD0D, $D3E5, and $D3E7 with pointers to its input routines and the FLEX input vector at $D3F7 with a pointer to its terminal status routine. However, it copies the low-level FLEX vectors at $D3E5 thru $D3FC for its use before replacing them, so it can use the original low-level terminal input and status vectors to avoid the necessity of containing its own terminal drivers. The only device-specific code in the typeahead routine turns interrupts on in the ACIA.

Since the code which sets up the ACIA for interrupts, establishes the circular queue, and modifies the FLEX vectors is needed only once, it is placed into the same area as used by the circular queue to save space.

PROBLEMS

The primary problems with the use of this or any other typeahead routine under FLEX

involve application programs which do not use the FLEX vectors for terminal input and status functions and those which turn interrupts off more than briefly. Typically, the programs which avoid the FLEX functions either drive the interface device directly or call SBUG or other monitor routines. These programs must be modified to work correctly with typeahead.

There are several low-level FLEX vectors which are not defined in the usual FLEX manuals. The ones of most interest here are as follows:

```
$D3E5 input from terminal
$D3D7 determine terminal status
$D3D9 output character to terminal
$D3DB input from terminal and echo it
```

Note that these are addresses, not instructions, and must be referenced indirectly, as in "JSR [$D3E5]". By modifying the programs to use these vectors, they will be made compatible with the use or non-use of the typeahead routine, and they will be more portable.

If a program which suppresses interrupts more than momentarily cannot be easily modified to allow them, it must be modified to use its own terminal input and status routines in order to be compatible with the typeahead routine, or the program may be used only in the absence of the typeahead routine.

TYPEAHEAD PROGRAM

The program below, when executed, will link itself into the appropriate FLEX locations and vectors. It is assumed to be loaded into the end of a 2048 byte RAM located at $E800 to $EFFF. As it stands, it requires 378 bytes, including a 256 byte circular buffer.

```
*
* FLEX typeahead program
*
port    equ   $e004      input port
warms   equ   $cd03      flex return
ichv    equ   $cd0e      flex input vector
reset   equ   %00000011  master reset for acia
prtini  equ   %11010101  initialization for acia
vector  equ   12         flex vector count
flxvec  equ   $d3e5      low-level flex vectors
input   equ   $00        vector table ofsets
ihandl  equ   $02
status  equ   $12
output  equ   $14
inche   equ   $16
*
        org   $ee80
*
start   bra   irq
*
incnt   fcb   $00        queue count
inptr   fdb   initiz     queue input pointer
ouptr   fdb   initiz      queue output pointer
vecsav  rzb   vector<<1  flex i/o vector save area
*
*       handle interrupts
*
irq     pshs  cc,d,dp,x,y,u save registers
        jsr   [vecsav+status] get status
        beq   contin     skip if no input
        jsr   [vecsav+input] get the character
        ldb   incnt      bump character counter
        incb
        bne   notful     warn if queue full
        lda   #$07
        jsr   [vecsav+output] beep user
        bra   contin     do not store character
notful  stb   incnt
        ldx   inptr      point to buffer queue
        sta   ,x+        store character
        cmpx  #initiz+256 end of queue ?
        blo   circi
        ldx   #initiz    reset buffer pointer
circi   stx   inptr
contin  puls  cc,d,dp,x,y,u
irqsav  jmp   irqsav     continue interrupt chain
*
*       getchne gets char from input queue without echo
*
getchne pshs  cc,b,dp,x,y,u
test    bsr   stats      chars in the buffer ?
        beq   test
        ldx   ouptr      get pointer to buffer
        lda   ,x+        get character
        dec   incnt      bump count down
        cmpx  #initiz+256 end of queue ?
        bls   circo
        ldx   #initiz    reset the pointer
circo   stx   ouptr      and save it
        puls  cc,b,dp,x,y,u,pc return
*
*       getche gets character from queue and echoes it
*
getche  bsr   getchne    get a character
        pshs  cc,d,dp,x,y,u
        jsr   [vecsav+output] echo it
        puls  cc,d,dp,x,y,u,pc return
*
*       test input queue length
*
stats   tst   incnt
        rts
*
*       initialize vectors, set interrupts, fix flex vectors
*
initiz  pshs  cc         save condition codes
        orcc  #$50       turn off interupts
        ldx   #flxvec    save flex i/o vectors
        ldy   #vecsav    save area
        lda   #vector    count
loadct  ldu   ,x++
        stu   ,y++
        deca
        bne   loadct
        ldx   flxvec+ihandl save flex irq vector
        stx   irqsav+1   in jump at end of irq
        ldx   #irq       change flex irq pointer
        stx   flxvec+ihandl
        ldx   #getchne   change flex vectors
        stx   flxvec+input
        ldx   #stats
        stx   flxvec+status
        ldx   #getche
        stx   flxvec+inche
        stx   inchv
        stx   inchv+3
        bsr   iniprt     initialize control port
        puls  cc         restore condition codes
        andcc #$ef       enable interrupts
        jmp   warms      return to flex
*
*       initialize control port: change if not acia
*
iniprt  lda   #reset     master reset acia
        sta   port
        lda   #prtini    reinitialize with interrupts
        sta   port
        rts
*
        rzb   256-(*-initiz) rest of buffer
*
        end   initiz
```

COBOL Name & Address System

The diskette accompanying this letter contains a name and address system I put together using Crunch COBOL. It uses both the CALL CHAIN and the CALL OVERLAY verbs, and demonstrates a mechanism for using direct cursor addressing for terminal output. The terminal I am using is an old Xitex SCT-100 (real old - 300 baud). The cursor routines can be easily modified to drive any ASCII terminal - for instance, the delay loops would need to be adjusted or removed for terminals faster than the SCT-100.

The programs that make up this system are:

NAMP0000 - Initialize file.
NAMP0001 - Menu.
NAMP0002 - Build screen for ADD function.
NAMP002A - ADD records to file.
NAMP0003 - Build screen for INQUIRE function.
NAMP003A - INQUIRE on existing records.
NAMP003B - Re-initialize INQUIRE screen.
NAMP0004 - Build screen for UPDATE function.
NAMP004A - UPDATE existing records.
NAMP004B - Re-initialize UPDATE screen.

To run the system, first execute NAMP0000. This program will ask for the size of the file you wish to initialize (number of records). NAMP0000 can also be run from the menu by selecting option 'F' (this option does not appear on the menu). After initializing the file, control will be transferred to the menu. The options on the menu are pretty much self-explanatory.

To back up a field when adding or updating a record, enter the character '<' in the field the cursor appears in and press return. The cursor will then back up and allow corrections to be made.

To return to the menu from the ADD function, enter the word 'END' in the first name field. When in the UPDATE function, enter 'END' in the last name field. The INQUIRE function displays a message at the bottom of the screen after displaying a record. Enter 'E' to return to the menu from here.

The UPDATE function allows fields to be blanked by entering the character '~' in the field. Entering a '#' will skip the remaining fields on the screen. The system will ask for a confirmation before updating the record. Enter a 'U' to update the record, a 'D' to delete the record, or a 'C' to continue searching the file.

The programs NAMP003B and NAMP004B are overlays to NAMP003A and NAMP004A respectively. The reason for the overlays is that I had problems compiling the programs - apparently the programs were to big. I believe the system would probably work a little better if these were changed to be linked programs, but I just haven't gotten around to making the required changes. If anyone is interested, I could make these changes and provide standard cursor control support (VT, HT, BS, LF, FF) for most all terminals. Drop me a line at the above address and we'll discuss it.

I hope this system or some parts of it will prove useful to some of your readers. If anyone else is interested in developing applications in COBOL for 5550 machines, I would love to hear from them.

Mike Martin

CRUNCH COBOL - a sleeper ZZZzzzzz

This COBOL thing has caught me off guard. First, I don't program in COBOL worth a hoot. Second, I find out a lot of you do. Thirdly, our CRUNCH COBOL is a 'hotter' number than I even imagined. I don't know if it is the one half (1/2) price we (S.E. MEDIA) are selling CRUNCH COBOL at or that a lot of you have just been waiting for a good COBOL sub-set. Anyway, it is one of our hotter sellers. And even better, I get a lot of nice compliments about it. Guess what, I'm going to take home a COBOL package and try to learn something, don't wanna be left out!

So, here is a COBOL program written in CRUNCH COBOL. We will run this as a short series - this is part ONE. I have promises from others for more COBOL articles. Let's see how you like this one. As a tutorial it alone is worth it. As a working applications - it is. So, have at it - and if you havn't jumped into COBOL, on your 550 6809 system yet, well here is your chance. Also I have instructed Bob, Chris and Tom, over at S.E. MEDIA, to hold the one half special on for a while longer. And, if you include an additional $9.95, I will send you a 5 or 8 inch disk with all these COBOL programs on it so you do not have to type them all in. If you have already bought Crunch COBOL from S.E. MEDIA, let us know your purchase date and for $9.95 you can get these source files on disk - make sure you tell them what name it was purchased under. So, **START LEARNING AND ENJOYING A CLASSIC LANGUAGE - COBOL.**

I wonder if we are becoming a 'languages' magazine?

DMW

```
IDENTIFICATION DIVISION.
PROGRAM-ID. NAMP0000.
ENVIRONMENT DIVISON.
FILE-CONTROL SECTION.
    SELECT NAME-FILE ASSIGN TO "WNAMD0001DAT"
           ACCESS RANDOM.
DATA DIVISON.
FILE SECTION.
FD  NAME-FILE.
01  NAME-RECORD      PIC X(126).
WORKING-STORAGE SECTION.
01  WS-TITLE     PIC X(27)
        VALUE    "NAME/ADDRESS FILE FORMAT ".
01  WS-RESPONSE PIC X(01).
01  FILE-SIZE       PIC 9(4) VALUE ZEROES.
    COPY "WWFCCDT00CBL".
    COPY "WNAMCDMSTCBL".
PROCEDURE DIVISION.
    COPY "WWFCCPT01CBL".
A000-INITIAL.
    MOVE 4 TO X.
    MOVE 5 TO Y.
    PERFORM Z110-ACA THRD Z110X.
    DISPLAY I-O "ENTER FILE SIZE " UPON CONSOLE.
    ACCEPT FILE-SIZE FROM CONSOLE.
    IF FILE-SIZE > 1000
        MOVE 10 TO X
        MOVE 5 TO Y
        PERFORM Z110-ACA THRU Z110X
        DISPLAY I-O "FILE SIZE LIMITED TO 1000 RECORDS"
        MOVE 1000 TO FILE-SIZE.
    IF FILE-SIZE < 1
        MOVE 10 TO X
        MOVE 5 TO Y
        PERFORM Z110-ACA THRU Z110X
        DISPLAY I-O "FILE SIZE DEFAULTED TO 100 RECORDS"
            UPON CONSOLE
        MOVE 100 TO FILE-SIZE.
    OPEN OUTPUT NAME-FILE.
    MOVE SPACES TO WS-NAME-ADDRESS-RECORD.
    PERFORM A100-NEW-FILE THRU A100X FILE-SIZE TIMES.
    CLOSE NAME-FILE.
    PERFORM Z090-CLEAR-SCREEN THRU Z090X.
    DISPLAY "FILE FORMATTED" UPON CONSOLE.
    CALL "NAMP0001" CHAIN.
A100-NEW-FILE SECTION.
    MOVE "@" TO WS-NAR-EOR.
    WRITE NAME-RECORD FROM WS-NAME-ADDRESS-RECORD.
A100X. EXIT.
    COPY "WWFCCPT00CBL".

IDENTIFICATION DIVISION.
PROGRAM-ID. NAMP0001.
ENVIRONMENT DIVISON.
DATA DIVISON.
WORKING-STORAGE SECTION.
01  WS-TITLE     PIC X(27)
        VALUE    "NAME/ADDRESS MAINTENANCE ".
01  WS-RESPONSE PIC X(01).
    COPY "WWFCCDT00CBL".
PROCEDURE DIVISION.
    COPY "WWFCCPT01CBL".
    MOVE 5 TO X.
    MOVE 22 TO Y.
```

```
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "A - Add record" UPON CONSOLE.
    MOVE 7 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "I - Inquiry" UPON CONSOLE.
    MOVE 9 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "U - Update" UPON CONSOLE.
    MOVE 11 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "E - Exit system" UPON CONSOLE.
    MOVE 13 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "Enter selection- _" UPON CONSOLE.
    PERFORM Z040-BACK-SPACE-CURSOR THRU Z040X.
A100-RESPOND.
    ACCEPT WS-RESPONSE FROM CONSOLE.
    IF WS-RESPONSE = "E"
        PERFORM Z090-CLEAR-SCREEN THRU Z090X
        STOP RUN.
    IF WS-RESPONSE = "A"
        CALL "NAMP0002" CHAIN.
    IF WS-RESPONSE = "I"
        CALL "NAMP0003" CHAIN.
    IF WS-RESPONSE = "U"
        CALL "NAMP0004" CHAIN.
    IF WS-RESPONSE = "F"
        CALL "NAMP0000" CHAIN.
    MOVE 15 TO X.
    MOVE 15 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "INVALID SELECTION - ENTER A, E, I, OR U"
        UPON CONSOLE.
    MOVE 13 TO X.
    MOVE 39 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    GO TO A100-RESPOND.
    COPY "WWFCCPT00CBL".


IDENTIFICATION DIVISION.
PROGRAM-ID. NAMP0002.
ENVIRONMENT DIVISON.
DATA DIVISON.
WORKING-STORAGE SECTION.
01  WS-TITLE     PIC X(27)
        VALUE    "NAME/ADDRESS RECORD ADDS ".
    COPY "WWFCCDT00CBL".
PROCEDURE DIVISION.
    COPY "WWFCCPT01CBL".
    MOVE 1 TO Y.
    MOVE 5 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "FIRST NAME:"
        UPON CONSOLE.
    MOVE 34 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "LAST NAME:"
        UPON CONSOLE.
    MOVE 7 TO X.
    MOVE 1 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "ADDRESS 1:"
        UPON CONSOLE.
    MOVE 34 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "ADDRESS 2:"
        UPON CONSOLE.
    MOVE 9 TO X.
    MOVE 1 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "CITY:" UPON CONSOLE.
    MOVE 24 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "STATE:" UPON CONSOLE.
    MOVE 34 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "ZIP:" UPON CONSOLE.
    MOVE 49 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "TEL:" UPON CONSOLE.
    MOVE 11 TO X.
    MOVE 2 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "XMAS:" UPON CONSOLE.
    MOVE 12 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "PARTY:" UPON CONSOLE.
    MOVE 13 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "BUSINESS:" UPON CONSOLE.
    MOVE 14 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "BILL:" UPON CONSOLE.
    CALL "NAMP002A" CHAIN.
    COPY "WWFCCPT00CBL".


IDENTIFICATION DIVISION.
PROGRAM-ID. NAMP0002.
ENVIRONMENT DIVISON.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT NAME-FILE ASSIGN TO "WNAMD0001DAT"
        ACCESS MODE IS RANDOM
        FILE STATUS IS NAME-STAT.
DATA DIVISON.
FILE SECTION.
FD  NAME-FILE.
01  NAME-RECORD PIC X(126).
WORKING-STORAGE SECTION.
01  WS-TITLE     PIC X(27)
        VALUE    "NAME/ADDRESS MAINTENANCE ".
01  WS-RESPONSE PIC X(01).
01  REC-NO      PIC 9(2) VALUE 0.
    COPY "WWFCCDT00CBL".
    COPY "WNAMCDMSTCBL".
PROCEDURE DIVISION.
    OPEN I-O NAME-FILE.
    MOVE ALL "Z" TO WS-NAR-FIRST-NAME.
    PERFORM R100-READ-FILE THRU R100X
        VARYING REC-NO FROM 1 BY 1
          UNTIL WS-NAR-FIRST-NAME < "A" OR
                WS-NAR-FIRST-NAME = "END".
    PERFORM A000-ADD-RECS THRU A000X.
    CLOSE NAME-FILE.
    CALL "NAMP0001" CHAIN.
A000-ADD-RECS SECTION.
    PERFORM A100-PROCESS-LOOP THRU A100X
        UNTIL WS-NAR-FIRST-NAME = "END".
A000X. EXIT.
A100-PROCESS-LOOP.
    PERFORM C100-CLEAR-FIELDS TRRU C100X.
A101-FIRST.
    MOVE 5 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-FIRST-NAME FROM CONSOLE.
    IF WS-NAR-FIRST-NAME = "END"
        GO TO A100X.
A102-LAST.
    MOVE 5 TO X.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-LAST-NAME FROM CONSOLE.
    IF WS-NAR-LAST-NAME = "<"
        GO TO A101-FIRST.
A103-ADD1.
    MOVE 7 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-LINE1 FROM CONSOLE.
    IF WS-NAR-LINE1 = "<"
        GO TO A102-LAST.
A104-ADD2.
    MOVE 7 TO X.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-LINE2 FROM CONSOLE.
    IF WS-NAR-LINE2 = "<"
        GO TO A103-ADD1.
```

```
A105-CITY.
    MOVE 9 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-CITY FROM CONSOLE.
    IF WS-NAR-CITY = "<"
        GO TO A104-ADD2.
A106-STATE.
    MOVE 9 TO X.
    MOVE 31 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-STATE FROM CONSOLE.
    IF WS-NAR-STATE = "<"
        GO TO A105-CITY.
A107-ZIP.
    MOVE 9 TO X.
    MOVE 39 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-ZIP FROM CONSOLE.
    IF WS-NAR-ZIP = "<"
        GO TO A106-STATE.
A108-TEL.
    MOVE 9 TO X.
    MOVE 54 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-PHONE FROM CONSOLE.
    IF WS-NAR-PHONE = "<"
        GO TO A107-ZIP.
A109-XMAS.
    MOVE 11 TO X.
    MOVE 12 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-XMAS FROM CONSOLE.
    IF WS-NAR-XMAS = "<"
        GO TO A108-TEL.

A110-PARTY.
    MOVE 12 TO X.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-PARTY FROM CONSOLE.
    IF WS-NAR-PARTY = "<"
        GO TO A109-XMAS.
A111-BUSINESS.
    MOVE 13 TO X.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-BUSINESS FROM CONSOLE.
    IF WS-NAR-BUSINESS = "<"
        GO TO A110-PARTY.
A112-BILL.
    MOVE 14 TO X.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NAR-BILL FROM CONSOLE.
    IF WS-NAR-BILL = "<"
        GO TO A111-BUSINESS.
A199-CONFIRM.
    MOVE 16 TO X.
    MOVE 1 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "Enter 'Y' to confirm add. "
        UPON CONSOLE.
    ACCEPT WS-RESPONSE FROM CONSOLE.
    IF WS-RESPONSE = "Y"
        NEXT SENTENCE
    ELSE
        IF WS-RESPONSE = "<"
            GO TO A112-BILL
        ELSE
            GO TO A100X.
    WRITE NAME-RECORD FROM WS-NAME-ADDRESS-RECORD.
    ADD 1 TO REC-NO.
    PERFORM R100-READ-FILE THRU R100X
        VARYING REC-NO FROM REC-NO BY 1
          UNTIL WS-NAR-FIRST-NAME < "A".
A100X. EXIT.
C100-CLEAR-FIELDS SECTION.
    MOVE 5 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 15 TIMES.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 15 TIMES.
    MOVE 7 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 20 TIMES.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 20 TIMES.
    MOVE 9 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 10 TIMES.
    MOVE 31 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 2 TIMES.
    MOVE 39 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 9 TIMES.
    MOVE 54 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 10 TIMES.
    MOVE 12 TO Y.
    MOVE 11 TO X.
    PERFORM C300-CLEAR-FLAGS THRU C300X 4 TIMES.
    MOVE 16 TO X.
    MOVE 1 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM Z020-EOL-ERASE THRU Z020X.
C100X. EXIT.
C200-DISPLAY-DASH.
    DISPLAY I-O "_" UPON CONSOLE.
C200X. EXIT.
C300-CLEAR-FLAGS.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X.
    ADD 1 TO X.
C300X. EXIT.
R100-READ-FILE SECTION.
    READ NAME-FILE INTO WS-NAME-ADDRESS-RECORD
        KEY IS REC-NO
            INVALID KEY
              PERFORM Y100-FILE-ERROR THRU Y100X.
R100X. EXIT.
Y100-FILE-ERROR SECTION.
    MOVE 16 TO X.
    MOVE 1 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "FILE ERROR " NAME-STAT
                " RECORD # " REC-NO
                " <CR> TO CONTINUE"
        UPON CONSOLE.
    ACCEPT WS-RESPONSE FROM CONSOLE.
    MOVE "END" TO WS-NAR-FIRST-NAME.
Y100X. EXIT.
    COPY "WWPCCPT00CBL.".

IDENTIFICATION DIVISION.
PROGRAM-ID. NAMP0003.
ENVIRONMENT DIVISON.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT NAME-FILE ASSIGN TO "WNAMD0001DAT"
        ACCESS IS RANDOM
        FILE STATUS IS NAME-STAT.
DATA DIVISON.
FILE SECTION.
FD  NAME-FILE.
01  NAME-RECORD PIC X(126).
WORKING-STORAGE SECTION.
    COPY "WNAMCDINQCBL.".
    COPY "WWPCCDT00CBL".
    COPY "WNAMCDMSTCBL".
PROCEDURE DIVISION.
    COPY "WWPCCPT01CBL".
    MOVE 1 TO Y.
    MOVE 5 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "FIRST NAME:"
        UPON CONSOLE.
    MOVE 34 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY I-O "LAST NAME:"
```

```
        UPON CONSOLE.
    MOVE 7 TO X.
    MOVE 1 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 "ADDRESS 1:"
        UPON CONSOLE.
    MOVE 34 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 "ADDRESS 2:"
        UPON CONSOLE.
    MOVE 9 TO X.
    MOVE 1 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 "CITY:" UPON CONSOLE.
    MOVE 24 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 "STATE:" UPON CONSOLE.
    MOVE 34 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 "ZIP:" UPON CONSOLE.
    MOVE 49 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 "TEL:" UPON CONSOLE.
    MOVE 11 TO X.
    MOVE 2 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 "XMAS:" UPON CONSOLE.
    MOVE 12 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 "PARTY:" UPON CONSOLE.
    MOVE 13 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 "BUSINESS:" UPON CONSOLE.
    MOVE 14 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 "BILL:" UPON CONSOLE.
    OPEN I-O NAME-FILE.
    CALL "NAMP003B" OVERLAY.
    COPY "WWFCCPT00CBL".
IDENTIFICATION DIVISION.
PROGRAM-ID. NAMP003A.
ENVIRONMENT DIVISON.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT NAME-FILE ASSIGN TO "WNAMD0001DAT"
        ACCESS MODE IS RANDOM
        FILE STATUS IS NAME-STAT.
DATA DIVISON.
FILE SECTION.
FD  NAME-FILE.
01  NAME-RECORD PIC X(126).
WORKING-STORAGE SECTION.
    COPY "WNAMCDUPDCBL".
    COPY "WWPCCDT00CBL".
    COPY "WNAMCDMSTCBL".
PROCEDURE DIVISION.
A000-INQUIRY-CONTROL SECTION.
    ACCEPT WS-LAST-NAME FROM CONSOLE.
    IF WS-LAST-NAME = "END"
        CLOSE NAME-FILE
        CALL "NAMP0001" CHAIN.
    MOVE ZERO TO WS-REC-NO.
    PERFORM A100-INQUIRY-LOOP THRU A100X
        UNTIL WS-NAR-LAST-NAME = "1".
    MOVE SPACES TO WS-NAME-ADDRESS-RECORD.
    CALL "NAMP003B" OVERLAY.
A000X. EXIT.
A100-INQUIRY-LOOP SECTION.
    MOVE 0 TO EOF-FLAG.
    PERFORM R100-READ THRU R100X
        VARYING WS-REC-NO FROM WS-REC-NO BY 1
          UNTIL WS-NAR-LAST-NAME = WS-LAST-NAME.
    IF EOF-FLAG = 1
        GO TO A100X.
    PERFORM B000-DISPLAY THRU B000X.
A101-ASK.
    MOVE 16 TO X.
    MOVE 1 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM Z020-EOL-ERASE THRU Z020X.
    DISPLAY 1-0 "`C'=continue, `I'=new inquiry, `E'=end "
        UPON CONSOLE.
    ACCEPT WS-RESPONSE FROM CONSOLE.
    IF WS-RESPONSE = "C"
        MOVE SPACES TO WS-NAR-LAST-NAME
        GO TO A100X.
    IF WS-RESPONSE = "I"
        MOVE "1" TO WS-NAR-LAST-NAME
        GO TO A100X.
    IF WS-RESPONSE = "E"
        CALL "NAMP0001" CHAIN.
    GO TO A101-ASK.
A100X. EXIT.
B000-DISPLAY SECTION.
    MOVE 5 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-FIRST-NAME UPON CONSOLE.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-LAST-NAME UPON CONSOLE.
    MOVE 7 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-LINE1 UPON CONSOLE.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-LINE2 UPON CONSOLE.
    MOVE 9 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-CITY UPON CONSOLE.
    MOVE 31 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-STATE UPON CONSOLE.
    MOVE 39 TO T.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-ZIP UPON CONSOLE.
    MOVE 54 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-PHONE UPON CONSOLE.
    MOVE 11 TO X.
    MOVE 12 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-XMAS UPON CONSOLE.
    MOVE 12 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-PARTY UPON CONSOLE.
    MOVE 13 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-BUSINESS UPON CONSOLE.
    MOVE 14 TO X.
    PERFORM Z110-ACA THRU Z110X.
    DISPLAY 1-0 WS-NAR-BILL UPON CONSOLE.
B000X. EXIT.
R100-READ SECTION.
    READ NAME-FILE INTO WS-NAME-ADDRESS-RECORD
        KEY IS WS-REC-NO
            INVALID KEY
            MOVE 1 TO EOF-FLAG
            MOVE "1" TO WS-LAST-NAME
            MOVE "1" TO WS-NAR-LAST-NAME.
R100X. EXIT.
Z000-TERMINAL-CONTROL SECTION.
Z001-DELAY.
    ADD 1 TO DELAY-VAR.
Z001X. EXIT.
Z020-EOL-ERASE.
    DISPLAY 1-0 $05 UPON CONSOLE.
    MOVE ZEROES TO DELAY-VAR.
    PERFORM Z001-DELAY THRU Z001X
        UNTIL DELAY-VAR > 600.
Z020X. EXIT.
Z110-ACA.
    DISPLAY 1-0 $1B $3D
        ACA-ADDRESS (X) ACA-ADDRESS (Y)
        UPON CONSOLE.
Z110X. EXIT.
```

To be completed next month.

OS-9 SETIME Module

THE TIME IS NOW

by Oral E. Groves

The SETIME module that comes with OS-9 is not very convenient to use. It requires you to enter the complete date and time even if all you want to do is move the time ahead one hour. If you are exacting about the time (as I am) you get very annoyed by SETIME defaulting everything to zero. I can't think of any situation when I would want to set the date to zero and the times when I am setting my clocks to exactly 00:00:00 are pretty rare too. As long as I was using the OS-9 CLOCK module supplied with the computer, I didn't do anything about it. When I wanted to write a CLOCK module for my Robertson Electronics CLK 68-1 SS30 board I decided to do something about SETIME too.

Since I am so exacting about the time (I know the number of the National Bureau of Standards by heart) I didn't like the way the software clock module arbitrarily threw away clock ticks while the floppies were in use. I don't turn my system off very much and by the end of a week I could be several minutes slow. This was intolerable. I had a CLK 68-1 board from Robertson Electronics. There is an adjustment on the crystal to set it very precisely. The accuracy is satisfactory even to my exacting standards. Now all I needed was a software CLOCK module for OS-9. There wasn't one at the time I bought the board.

I had just retired the old SWTPC 6800 system. OS-9 was still new to me. Writing the clock module looked like a good way to learn OS-9. The 6809 assembly language didn't look too bad. I had been writing assembler 6800 for years. OS-9 had things called "system calls" that looked very strange to a guy used to JMPing right into FLEX. I had some OS-9 practice while writing the virtual disk program ("TURBO": '68 MICRO, Feb '85,p24.) This looked like a good way to get some more.

Compared to the program to set it, the CLOCK module itself was quite easy to write. All it had to do was initialize the PIA and read the clock at regular intervals. For this I could borrow from the 6800 FLEX version of the software. Unfortunately I still couldn't CHANGE the time. The clock is set by shifting a string of digits into it. Getting the date/time string into the correct format from the command line turned out to be the hardest. I finally wrote up the time setting code into a little module named TODset, which I could call from someplace and hand it a completed digit string. The first main program I tried just linked to this program named TODset and passed it a predetermined string. I could change the string using DEBUG and execute it to get the time set. That was last September. I had started the CLOCK module in May of 1984 and it still hadn't been telling the correct time. I used my kluge main program twice. In September I used it to correct the clock which was off several minutes in the two years since it had been last set. I used it again in October to change back to Central Standard time. It is now May, 1985. Setime was finished just in time to switch to Daylight Savings.

A project that lasts over a year should LOOK bigger. The three modules that make up the project only total 1200 lines of assembler. I confess it isn't the only thing I've done for a year (I have to spend a little time on software testing at the office for which I get paid enough to keep the wolf from repossessing my ELEKTRA.) On the other hand SETIME did present a chance to practice all the great programming techniques I had learned from '68 MICRO and other places. I eagerly attacked it with all the weapons in my arsenal. I flow-charted. I pseudo- coded. I modularized. I even built Greg Morse's memory and register dump code into it. I carefully tested each subroutine by itself and fixed the bugs one by one. Greg's routines made it very easy to find where things were going wrong. Finally the big day came. I assembled the whole thing. I tried it. It WORKED!

The version of the program I present here contains two conditional parameters. "Debug" invokes the code to call Greg Morse's dump routines which I keep in a source library. The other is "Rbtsn" which invokes the code to set the time on the Robertson Electronics CLK 68-1. If "Rbtsn" equals zero SETIME uses the OS-9 FSSTIM call and can be used on any system. If set to not-zero, then the parameter string is converted to the proper format for the TODset module. SETIME tries to f$link to this module and failing that tries to f$load it from the default execution directory. If everything is okay OS-9 returns the starting address in the Y register. TODset ends with a RTS instruction. It is called with a "JSR ,Y". For other hardware clocks it should only be necessary to change two subroutines in the program. "Convert" changes the current system time to TODset format. "Doset" calls the TODset routine. Also if the hardware clock in question doesn't support the day of the week, the "parse" subroutine will need a change.

I will send you a copy of the source if I get a OS-9 formatted 5-1/4 inch disk in a reusable mailer and $2.00 beer money. I can handle 35 or 40 tracks, single or double density, single or double side. Oh yes, also Tandy Color Computer OS-9 format. All correspondence should go to:

OE Groves
10207 Gillette
Lenexa Kan. 66215

OS-9 is a trademark of Microware Systems Corp; Elektra is a trademark of AAA Chicago Computer Center; SWTPc 6800 is a trademark of Southwest Technical Products Corp; CLK 68-1 is a trademark of Robertson Electronics; FLEX is a trademark of Technical Systems Consultants; TRS-80 Color Computer is a trademark of Tandy Corp.

DOCUMENTATION

Setime has improved syntax and expanded options which make it much more "friendly" to the user. The syntax of setime is YY/MM/DD,HH:MM:SS<CR>. The program will reset the time to:

Year	=	YY
Month	=	MM
Day of mnth	=	DD
Hour	=	HH
Minute	=	MM
Second	=	SS

Setime will accept any sensible string to set part of the date/time. For example: ,/5/59 will set just the minutes and seconds. The current system time is read and only the specified CHANGES are written back. NOTE: commas and spaces separate substrings and any non alpha-numeric will separate digits.

```
*       This program picks up a date/time
* string from the command line if there is
* one.  The clock board is initialized,
* interrupts are set to 32 per second and the
* FSTIME system call is set up.  If the
* command line was empty the system date/time
* is displayed and a request is issued for
* update.  If the response is a carriage
* return the program exits.  If the user
* responds with a new date/time string of the
* form: DOW,YY-MM-DD,hh:mm:ss<cr> the clock
*will be reset to:
*
* day of week     = DOW  (only first two
*                         letters are needed)
*       year      = YY
*       month     = MM
* day of mnth     = DD
```

```
*       hour       = hh
*       minute     = mm
*       second     = ss
*
* Setime will accept any sensible string to
* set part of the time and date.  For example:
* ,,5-59<cr> will set just the minutes and
* seconds.  NOTM: commas or spaces separate
* substrings and any non alpha-neric will
* separate digits.
*
 nam setime
 ifp1
 OPT -L
 use /d1/elektra_defs/os9defs
 endc
 ttl set system time and date module
*
vers set 1 version number
type set prgrm+objct
nuline set 80 control input missing error
badinp set 61 illegal inout rmat error
badrng set 52 number out of range error
ops set reent+vers
*
* N O T E    This program may be used without
* the R@bertson board by setting "Rbtsn" to 0.
*
Rbtsn set 0
*
* This is the module definition section  * * *
*
* * * *   Inke debugging code
*
debug set 0
*
 mod endset,namset,type,ops,strset,memset
*
 ifgt debug
 use /d1/elektra_defs/debug.src
 endc
*
*
svd rmb 2
svx rmb 2
svy rmb 2
*
* * * decimal digits
*
tencnt rmb 1
huncnt rmb 1
*
* * * * *    counters for parse
*
stgsep rmb 1
numsep rmb 1
*
* set up system time packet
*
sysyy rmb 1
sysmth rmb 1
sysdd rmb 1
syshh rmb 1
sysmm rmb 1
syss rmb 1
*
dow rmb 1 day of week
flg rmb 1 time/date string is changed
*
* set up todset packet
*
todyy rmb 1
todmth rmb 1
todd rmb 1
dhh rmb 1
todmm rmb 1
todss rmb 1
*
todow rmb 1
dtmstr rmb 4 set up date/time str to stdout
dtmnr rmb 24
buff rmb 35 read from stdin
*
endat equ .
*
* reserve at leaft 200 bytes decimal r the
* stack and end on a page boundry
*
stack rmb ((endat+$1CB)&$FF00)-endat
memset equ .
*
namset fcs "setime"
todstr fcs "todset"
okstr fcc 'OK<cr> or '
dtmini fcc 'DOW'
oknr fcc ' YY-MM-DD,hh:mm:ss'
 fcb $d,$a+$80
daytib fcc 'SunMonTueWedThuFriSat'
 fcb vers
*
* * * *   end module definition
*
* This part of the program does the initalizing
*
* * * Initialze storagm area
*
init leax sysyy,u
 ldb #15 15 bytes in time packet
 clra
 deca  put hex FF in all time packets
ini1 sta ,x+
 decb
 bne ini1
 leay dtmini,pcr init D/T strng
ini2 lda ,y+
 sta ,x+
 bpl ini2
 rts
*
* * * *   end Initalization
*
*
* This part of the program gets the date/time
*
getime leax sysyy,u get addr of time packet
 os9 f$time
 sta dow save diy of week
 rts
*
* * * *   end of get current date/time
*
* This is the main program which calls all
* of the others
*
*
strset equ *
 std svd keep cmd line pointers
 stx svx
 sty svy
 bsr init initailize storage
*
* * *   conditionally invoke debug code
*
 ifeq debug-1
 lbsr shoregs
 pshs x,y,d
 leax sysyy,u
 ldy #(15+24)
 lbsr shomem
 puls x,y,d
 endc
*
 lbsr getime get system time
*
* * *   conditionally invoke debug code
*
 ifeq debug-1
 pshs x,y,d,cc
 lbsr shoregs
 leax sysyy,u
 ldy #(15+24)
 lbsr shomem
```

```
 puls x,y,d,cc
 endc
*
 bcc clkok
 cmpb #208 e$ilmd-clock module not there?
 bne setexit other problem
 leax todyy,u initialize clock module
 os9 f$stim
 bcs setexit
 clr dow
clkok bsr parse get new time from cmd line
*
* * *   conditionally invoke debug code
*
 ifeq debug-1
 pshs x,y,d,cc
 lbsr shoregs
 leax sysyy,u
 ldy #(15+24)
 lbsr shomem
 puls x,y,d,cc
 endc
*
 bcc linok
 cmpb #nuline empty line?
 bne setexit
 lbsr display show cur time & ask for change
 bcs setexit
 std svd
 stx svx
 bsr parse get response
*
* * *   conditionally invoke debug de
*
 ifeq debug-1
 pshs x,y,d,cc
 lbsr shoregs
 leax sysyy,u
 ldy #(15+24)
 lbsr shomem
 puls x,y,d,cc
 endc
*
 bcc linok
 cmpb #nuline no input?
 bne setexit error
 clrb  no error
 bra setexit
linok lbsr getime get cur time
 lbsr convert change to TODset format
*
* * *   conditionally invoke debug code
*
 ifeq debug-1
 lbsr shoregs
 pshs x,y,d
 leax sysyy,u
 ldy #(15+24)
 lbsr shomem
 puls x,y,d
 endc
*
 lbsr doset set the time
setexit os9 f$exit
*
* * * *  end of main program
*
* * * * * *
* This routine parses a date/time string of
* the form:
*
*          Dow,YY-MM-DD hh:mm:ss<cr>
*
*  into TODset format
*
*     INPUT
*     svx  = buffer pointer
*     dp,u = paramater area
*     svd  = number of characters in buff
*
*     OUTPUT
*     x    = end address of input string
*     dp,u = unchanged
*     d    = distroyed
*
*    [12,u]= date/time string of the form:
*          YY  mth DL  hh  mm  ss
*          HEX HEX BCD BCD BCD BCD
*     for use by the TODset program
*
*
*  parse calls:            parse is called by:
*          nxtch                        main
*          bcdhx
*          getnum
*          getdow
*
parse equ *
 clr stgsep
 clr numsep
 clr flg
 ldd svd
 ldx svx
*
* if the line is empty then rtn a nuline error
*
 cmpd #1
 bgt pa1
 comb set carry bit
 ldb #nuline
 rts
*
* else while (nxtch != EOF)
*
pa1 lbsr nxtch get next char. in buffer
*
* * *   conditionally invoke debug code
*
 ifeq debug-2
 pshs x,y,d,cc
 lbsr shoregs
 leax svd,u
 ldy #(15+24+35)
 lbsr shomem
 puls x,y,d,cc
 endc
*
*
 pshs cc keep carry bit
 cmpa #$d a! EOF?
 bne pa2
 clrb yes-go back
 puls a,pc
*
* if (nxtch != alpha-numeric) then CASE nxtch
*
pa2 ls cc recover carry bit
 bcc pa3 is char alph-num?
 cmpa #', nxtch = comma?
 bne pa4
 inc stgsep
 clr numsep
 lbra endwhl
pa4 cmpa #$20 nxtch = space?
 bne pa6
pa20 lbsr nxtch skip spaces
 cmpa #$20
 beq pa20
 leax -1,x
 inc stgsep count only one space
 clr numsep
 lbra endwhl
pa6 cmpa #$d nxtch = EOF
 bne pa7
 lbra inperr
pa7 inc numsep nxtch = anything else
 lbra endwhl
*
* else (nxtch = alpha-numeric) CASE stgsep
pa3 ldb stgsep
 ifne Rbtsn
```

```
 bne pa8 stgsep 0? - do Day of Week
 lbsr getdow
*
* * *   nditionally invoke debug code
*
 endc
 ifeq debug-2
 pshs x,y,d,cc
 lbsr shoregs
 leax svd,u
 ldy #(15+24+35)
 lbsr shomem
 puls x,y,d,cc
*
 endc
*
 ifne Rbtsn
*
 inc flg
 lbcs inperr
 lbra endwhl
pa8 decb
*
 endc
*
 bne pa9 at date string? - CASE numsep
 ldb numsep
 bne pa10 numsep =0? - do Year
 leay todyy,u
 ldb #$99
 lbsr getnum
*
* * *   conditionally invoke debug code
*
 ifeq debug-2
 pshs x,y,d,cc
 lbsr shoregs
 leax svd,u
 ldy #(15+24+35)
 lbsr shomem
 puls x,y,d,cc
 endc
*
```

To be Continued....

==========

Bit Bucket

ARKANSAS TELEPHONE COMPANY, INC.

W. H. McCALLIN, P. O. Box A
CLINTON, ARKANSAS 72031

Mr. Larry G. Williams, Executive Editor
68' Micro Journal
P.O. Box 849
Hixson, Tn. 37343

Dear Mr. Williams:

We are trying to locate a complete General Ledger package with all the related software for an OS-9 system which has been written for utility accounting. We emphasize utility as there is a very specific method of accounting required for utilities (Rural Electric Cooperatives and certain Telephone Companies) by the local and state regulatory commissions, Federal Communications (FCC) and Rural Electric Administration (REA).

We have a Hazelwood Computer Systems, Helix OS-9, Level II V 1.1, with 256 K, 20 megabyte Winchester and 5 3/4" floppy disk drives.

Ms. Christine Kocher of Computer Publishing suggested that maybe this letter could be published in your "letter to the editor" colus and maybe one of your readers might be of assistance.

Your assistance will be appreciated.

Very truly yours,

Joe U. Cunningham
Bookkeeper

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

Dr. Dobb's Journal

5/23/85

Don Williams
Computer Publishing Center
68 Micro Journal
5900 Cassandra Smith Rd
Hixson TN 37343

DMW,

Over seven years. Congratulations! It's been a stormy year or so for computer magazines, with the Softalk bunch folding and Ziff-Davis shooting its feet off, but a few old-timers are surviving the storm, including 68 Micro Journal, sticking it out with the kind of integrity that has been one of the best features of the true microcomputer pioneers. But as you note, you can get as lonely as a Maytag repair man. Lonely because, early on, the press flocked to the S100 crowd, and later it nattered about IBM and hovered over a former hobbyist company that has grown too big, forgotten its roots, lost its finest talent, misperceived its markets and locked a powerful microprocessor in a closed architecture machine with a paternalistic user interface. Lonely because books like Fire in the Valley don't tell the whole story. (I could argue that no book ever tells the whole story, and that - but that's all matter for another letter.)

But the people who saw the advantages of 6811 systems had no reason to be lonely, because, since 1978, they've had 68 Micro Journal, filled with tips and bug fixes, code and construction articles, smoking soapboxes and ramblings and such. Despite some idiosyncrasies of grammar you deliver the 6811 goods. You should be proud.

Mike

Mike Swaine
DDJ

M&T Publishing, Inc. ○ 2464 Embarcadero Way, Palo Alto, CA 94303 ○ 415 424-0600

Ed's Note: Thanks Mike, your nice letter has made our day. Dr. Dobb's was our 'role model' for the manner of handling authors and articles, when we first started. Reader participation seemed such a necessary part of a 'real' users forum.

I must say that Dr. Dobb's Journal has also remembered us even to this day. While general in nature we all get a big smile when we find one of your 68XX articles. Which is more than most all the others, besides us, are doing for 68XX users.

We are proud, Mike. Not just because we were able to hang in there. But because our readers, advertisers and others hung in there with us. And it is a very special kind of pride we feel, when I receive a letter, such as this very kind one from you. And I might add, "Thanks, I needed that." Again, you made our day!

All of us here at CPI, wish only the best for you and the crew there at Dr. Dobb's. You too fill a very special place that the others have left void. As this thing changes a lot of users and readers are beginning to understand that being big ain't everything! Have a GOOD one all!

DMW

GIMIX INC.
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

Introducing the GMX-020™ - MC68020 Processor Board from GIMIX

MC68020 32-Bit MicroProcessor

The GMX-020 CPU Board uses the state-of-the-art MC68020, the newest and most powerful member of Motorola's M68000 family of microprocessors. The MC68020 is a full 32-bit processor with separate 32-bit address and data buses, an on-chip instruction cache, and a coprocessor interface. The MC68020 is object-code compatible with earlier members of the M68000 family, with enhancements to the instruction set providing additional support for high-level languages and systems software. The processor also supports demand-paged virtual memory.

The pipelined internal architecture of the MC68020 allows overlapping execution of instructions, and can result in a net instruction execution time of zero under certain circumstances. This, along with the on-chip cache and other enhancements make the processor typically 400% more powerful than its predecessors. The 16 MHz version can process instructions at a sustained rate of 2 to 3 million instructions per second (MIPS) and at burst rates exceeding 8 MIPS.

GMX-020 Processor Board

The GMX-020 CPU Board is designed for maximum utilization of the power of the MC68020, while retaining compatibility with the already proven GIMIX line of peripherals such as DMA disk controllers and intelligent I/O processors. The board features:

- MC68020 processor operating at a 12.5 MHz clock rate (16.5 MHz optional when available)
- A 4K byte (1K long word) instruction-only physical address cache operating at full processor speed (no wait-states). The on-board cache can be operated in any one of four modes to optimize cache utilization for a particular operating system or application. The cache RAM can also be used as high-speed (no wait-state) RAM when the cache is not enabled.
- A high-speed, discrete Memory Management Unit (MMU) that supports multi-user, multi-tasking operation and demand-paged virtual memory environments. Use of the MMU causes no increase in memory access time. In addition to dynamic address translation, the MMU associates four separate attributes with each 4K segment of memory: a write-enable bit to protect shared text, a "valid" bit to flag segments containing valid data, a "dirty" bit to flag segments that have been modified, and an "access" bit to indicate that a segment has been used. The standard MMU configuration supports 4 Megabytes of virtual memory with up to 16 separate segment maps. Other configurations can allow up to 8 Mbytes of virtual memory, or up to 64 separate maps.
- An optional floating-point coprocessor (MC68881) that directly extends the architecture and the instruction set of the processor to include floating-point data types, full support for IEEE Rev 10.0 high level math functions, and also transcendental and other standard math functions. All coprocessor calculations are performed to 80 bits of precision.
- Six levels of prioritized autovector interrupts from seven sources. Two interrupt sources are internal to the board, three are from the bus, and two (non-maskable) are from sources connected directly to the CPU board.
- Three separate hardware prioritized channels for external DMA devices. Simultaneous DMA requests on different channels are arbitrated by the board on a channel priority basis.
- A 20-bit external address bus for up to 1 Megabyte of physical memory space (RAM and I/O). The I/O devices occupy the upper 4K bytes of the 1 MByte address space. Two separate areas are defined within the I/O space, each with optimum timing for particular I/O devices. (Note: The I/O timing will not support any 6800/6809 peripheral devices such as the 6850 or 6821. Serial and parallel I/O is supported only through GIMIX intelligent I/O processors.)
- Two EPROM sockets that accept 8K, 16K, 32K, or 64K x 8-bit industry standard devices for up to 128K bytes of on-board firmware. The EPROMs are addressed above the 1 MByte RAM space, with auto-mapping of the restart vectors to low memory on power-up or reset.
- A full-featured hardware time-of-day clock/calendar with battery backup, which can also generate interrupts at one second intervals.
- A separate "tick" generator that can generate interrupts at precise, jumper selectable intervals ranging from 10 microseconds to 20 minutes. Interrupts from the "tick" generator can be enabled or disabled under program control, and have their own priority level to minimize overhead during context switching.
- A separate voltage regulator board that powers the board and provides standby battery power for the TOD clock. The regulator board receives its input from the standard power supply in the GIMIX mainframe.

68 Micro Journal,
5900 Cassandra Smith Rd.,
Hixson, TN 37343

MICRONICS RESEARCH CORP
Microcomputer Hardware and Software
GIMIX® Sales, Service and Support

Dear Don,

Earlier this month I typed in Ben Slaghekke's DO utility, published in the January issue of 68MJ, which performed exactly as described. After playing with it a while, however, I discovered that DO would request keyboard input only if it found itself lacking a parameter, and then only if the following line contained a command (such as +CAT), which meant that such a KBD call had to be the final call in the command-file, and had to be followed by a CMD call, whether I needed it or not, otherwise DO would not exit gracefully. Also, not all simple keyboard-responses, such as Y or N, are necessarily followed by a CR, and DO did not ALWAYS handle this situation correctly.

Sometimes I needed the sequence p1, p2, KBD, KBD, p3, p4, +CMD, KBD, p5, KBD for example (where p1, etc., are parameters, and KBD indicates a KBD response), without having to end the sequence with a non-required +CMD in order for the final KBD to function correctly. The patch to accomplish this is quite simple, though it took me quite a while to realise that I had to distinguish between keyboard responses (usually strings) which require a final CR and those which do not. I chose the tilde and the backslash to symbolise these KBD responses, though you may select a different set of literals, of course.

Here are the changes required :

```
1. Add to the CUSTOMIZABLE CONSTANTS :
KBDONE EQU '~ Single-char from KBD
KBDSTR EQU '\ String response from KBD (last char=CR)

2. Repace PARAM1 through PARAM2 with the following :
PARAM1 PSHS B  (* delete the line LDB LSTCHR-BASE,U)
       TST PLUSFLG-BASE,U
       BNE PARAM2
       BSR ECHO
       TST LITERA-BASE,U
       BNE PARAM3
       CMPA #KBDONE Single KBD char?
       BEQ PARAM2   Yes
       CMPA #KBDSTR String input from KBD?
       BNE PARAM3   No
PARAM1 INC PLUSFLG-BASE,U Set KBD Flag
PARAM2 LBSR KBD1N
       CMPA #CR
       BNE PARAM3
       CLR PLUSFLG-BASE,U Reset KBD Flag
       JSR PUTCHR .........
```

Now my command-file for the GIMIX FORMAT.CMD consists of
```
       +FORMAT
       Y5DD40
       S2\\Y~
```

where the backslashes allow me to enter the Disk's Volume-Name and Volume-Number respectively and the tilde gives me my final chance to ABORT or not.

I suspect that the initial PSHS B and the final PULS B are unnecessary now that the LDB line is deleted, but I did not have time to check this out.

Your MAY issue was great ... causing me to write to at least 3 of your correspondents. Keep it up!

Sincerely,

33383 LYNN AVENUE,
ABBOTSFORD,
BRITISH COLUMBIA,
CANADA, V2S 1E2

(604) 859-7005

R. Jones
President

I am writing on an airplane after reading your June editorial and if I don't put my thoughts down now, they will surely be lost. I have read the journal for several years and appreciate your candor in the review of products which may adversely influence their advertising posture. I support these products and have never been disappointed with their performance.

I work for a firm which supplies industrial automation equipment to the semiconductor industry. These are motion control systems, employing pattern recognition techniques, that attach interconnecting wires (ultrasonically) between the chip and header. We are a heavy user of 68xxx components. Most of our work in the past has been based on applications of the 6809 as a controller. Our new product development activities have pushed us into more activity with the 68000. This entailed the evaluation of bus structures and available 68000 based hardware. We finally settled on a VME bus structure, a Force CPU, and a PDOS operating system. During this exercise I was exposed to a plethora of board products and attendant sales pitches.

Now to the point. VME bus hardware has been high priced, but it would appear that this situation is rapidly changing, and is competitive with current journal advertisers. I think coverage of the VME bus and these products would be of reader interest and lead to some advertising from folks like Force, Mizer, Ironics, etc. Also there is software support form Microware and Eyring Research. The VME User Group publicity committee might supply article material.

My mind is boggled by operating systems! Having to juggle half a dozen can be very confusing. However, I think an article on PDOS would be interesting. I know in my application (real time multi tasking) its a winner.

A comment I have seen from time to time in your letter publications is a plea for more hardware articles. I think the reason you don't get them is the lack of full page schematic support. I know several times I have had circuits which would be of interest, but the format puts me off. This could be more important than a fancy cover!

In conclusion, don't let the journal fade. It would leave a big hole.

Sincerely,

Chuck Sublett
175 S. Los Mesquitee
Orange, CA 92669

Ed'd Note: Chuck, thanks for the prompt reply. Seems like the June ramblings grabbed a lot of folks. I have never received as much response to any one editorial as that one. As for the VME bus and other related subjects, well, actually I just have not received much. How about it? Also I do not hear much interest, from our readers, concerning the VME or VERSABUS. Like you say it is typically Motorola priced - high. However, as interest grows (and Motorola prices decline to a reasonable level) we will cover more of each. Intel has had a much better pricing and support system. Heck, when we started with Motorola devices, Intel wasn't even around, wonder what happened?

Also as to full page schematics, we can spread them over two pages, if necessary. We have problems with something like a high density D or E size drawing, but if we get it in reasonable size, no sweat. Also you should see some of the really fine stuff that has come in here that we were unable to use because it was produced with everything from cuckle stick to crayons. We **DO NOT** redraw diagrams, that is any that entail probably more than three or four devices. If it is neatly done (reasonably so) and dark enough to reproduce photographically, then no problem, I don't care if it is printed on an old bed sheet or toilet paper. So, if you have something our readers might be interested in, **PLEASE** send it on! Just try to bear in mind the size of our page, which is normal magazine size.

DMW

Don,

I tried to check in to the 68XX net on 75 meters on 5/8/85. I heard about 10 stations, but I don't know if anyone heard me. I havn't heard anyone on Wednesday since. I have an original SWTPC 6800 running FLEX. I am interested in RTTY, Packet Radio and CW, but have nothing running now. My station is completely homebrew and would be very interested in RTTY programs. I will continue to listen around 3870 KHZ on Weds evenings.

73

Stan Houk AA3P
2343 Peach Tree Lane
Dyer, IN 46311

Editor's Note: Thanks Stan for the note above. Well, I was having a thunder and lightening storm right about the time I started calling. My static was steady 10-15 db over S9 and peaking up to 30 over. I could hardly hear anyone. I could hear some in there calling, but I couldn't get much better than a 2 to 3 copy.

Since then I have listened but the summer time conditions are here and I think 75 meters is about done for me until this fall. However, I have getting notes and calls from others who are able to get together there (about 3870 KHZ) each Wednesday. Some call there and then move off to QSO.

I will continue to listen but if not better than last few weeks, I will wait till fall to continue. However, you might continue to listen and let me know how things are progressing. I hope that once started, it will continue and group up according to interest. But anyway it goes will be fine, as long as those who desire can make it.

Best I can figure there were about 30 to 40 trying to get in but conditions that evening were of the worse type. The time of day wasn't too good either. Due to daylight saving time, it was still light here in Tennesse and as you go towards the west, it was lighter and worse (short skip). So let me know what you folks want to do and I will cooperate. Send in your comments and I will try to put something here as needed.

However, lets not give up. For those that can make it - KEEP IT UP. By fall you should have a good foundation established. If not sooner, see you then.

DMW

Dear Group:

This is a letter of recommendation for one of your advertisers. PERIPHERAL TECHNOLOGY has been an advertiser for a long time in the JOURNAL and I decided several months ago to get rid of the old 6809 box. I want you know that this required a great deal of soul searching to out an old friend away. I needed more room on the desk, so away it went! I still (once in awhile) bring it out to transfer some software that I decide to use again.

I bought two PT-69 systems, one for OS-9 and the other for FLEX. Both systems have operated from scratch without a hitch. The people who are just now getting around to the 68xx(x) world just don't know what that missed by working with the old SS-50 (&) systems. That remark is not to cast any disparaging remarks on current systems. The new systems are fantastic compared to old ones of 1975-1978 era. I never saw a single system with a cover in place. We were always tinkering with something.

Anyway, I have attached a modification that your readers who own a PT-69 may be interested in. It is a charging resistor and diode for battery backup on the clock if one uses nicads for power.

I have added a 390 ohm resistor with 1N4148 diode as shown. With two nicads (2.7 v total) the charging current is about 3.3 milliamperes. I found that with the power off to the system, the current drain was about 170 microamperes. This is caused by the 4.194304 MHZ clock crystal. A lower frequency (I'm told) will result in a current drain of only 50 microamperes. I did try three nicads which gave a voltage of about 4 volts and the current drain was over 320 microamperes! My recommendation is two nicads (don't use the charger is one uses regular batteries) with the charging circuit in place and one doesn't have to worry about it.

I used two PANASONIC nicads (they were $1.25 mail order from a hobby place) since they had solder tabs and were 500 MAH cells.

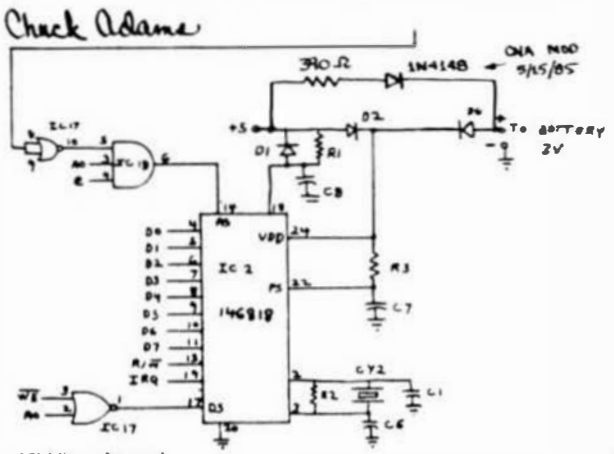
I hope this idea will help someone else. I am interested in starting at PT-69 users group and can do disks for $5.00 each until we get bigger. Please write, don't call, as I tend to loose notes taken at the phone. My apologies to those that I have lost and if I owe you, just drop me a line.

The Dallas group has been reborn (it never really died, just got low profile). Drop us a line if you live in the area and are interested in driving to GARLAND. It's over an hour drive one way for me and I don't miss a meeting.

Thanks,

Chuck Adams

Chuck Adams
Box 6809
Denton, Tx 76203
(817) 565-2821 (work)



10207 Gillette
Lenexa Kn. 66215

Dear Don:

I just finished the May issue of '68 Micro. As usual I couldn't put it down and also as usual I had a bunch of other stuff I should have been doing; but I was struck by your "Ramblings and Such." I am in the middle of another article for '68 Micro and your comments really hit home. I had the strangest feeling you were talking to ME. I got my SWTPC 6800 in 1978. I don't know if that qualifies me as an "old timer" in this business or not, but I feel the same way you do about our little "club". In this world of mass marketing the S50 and 68xxx fans all count. We are each responsible for the advancement of the group as a whole. Any one of us can either help or hinder. It is both wonderful and sad that an indivdual counts for so much. It hurts me to not see the old familiar AAA Chicago Computer ads in '68 Micro since last January. I retired the SWTPC to the hardware test bench last spring in favor of an AAA Electra and OS9. The combination is wonderful and I enjoyed the late night telephone calls and corespondance I had with Jerry Koppel while I tried to get all the pieces assembled. I may still have the only Elektra running OS9 AND a Computer Excellence Memory board. I feel vaguely uneasy about all the names that we heard and read about back then and now no longer do. Names like Percom, Midwest Scientific, and Ohio Scientific to mention just a few that had to finally give up and go find a bigger audience. I want to personally thank you and your loyal crew for sticking by us for the last seven-plus years and I want you to know I will do as much as I can to help you keep the show on the road. May the greatest of good fortune go with you.

Regards,

Oral E Groves

OE Groves

Ed's note: Oral, it is folks and attitudes like yourself and thoughts expressed in your letter, that keeps us going.

The worst part about it is that those you mentioned, with the exception of AAA Chicago (back with us), SWTPC, Computer Excellence and Ohio Scientific, didn't make it, they didn't understand where the market was going, in my opinion. Ohio Scientific started making the transition back in the days when it could still be done at relatively small cost, as compared to todays market entry level cost. They are still around. Percom and Midwest Scientific are gone, for all practical purposes. The rest you mentioned are alive and kicking! You must understand the market and act in time! We still have an excellent market window to sell into, if we use common sense and know what direction we are pointed. Poor management, and a lack of market analysis has done more to kill off some our older vendors and manufacturers than any other factors. It all led to smaller sales, less, then to minus cash flow and then ZAP, they were gone!

We, (I), have done some consulting over the past 5 or 6 years. I have had an above average success rate of calling the market. I feel that I am closer to my readers (who are users) than most publishers and survey analyst. We work in number that insure from 1 to 5 percent accuracy. Some of the largest corporations in the world have used my services, and paid nice fat sums for information. You would be surprised at how much information we developed on the CoCo market alone. Yet, at various times I gav the same essential information to about any of our advertisers that asked, for free. Over 90% of those who reacted accordingly have survived! Not that I am really all that smart, it is because I learn and listen to what YOU tell me! It was your input that gave us direction. THAT SIMPLE!

When the day comes that I think that I know more about you than you do yourself. That is the day my employees had better start looking for jobs elsewhere! So I thank each and ev ryone of you for the loyalty shown

us over the years, and look forward to many more years, with your support.

DMW

F.S. Also looking forward to the article mentioned in your letter. For that we all thank you and all those others who share their experiences and "fruits of love and labor" with all the rest of us - THANKS!

LEENSHIRE

24th May 1985

Leenshire Ltd
Moorside Road, Winnall
Winchester, Hants SO23 7RX
England

Tel: Winchester (0962) 64175
Telex: 477300 LSHIRE

Computer Publishing Center,
68' Micro Journal,
5900 Cassandra Smith Road,
Hixson,
TN 37343,
U.S.A.

Dear Sir,

We have recently implemented an OS9/68000 VME-Bus System, using FORCE SYS68K Boards. We have been using OS9/6809 for some time, and have many disc and ROM-based systems in the field.

As we still need to produce software for these 6809 systems (some under OS9, some stand-alone), we require a relocatable 6809 cross-assembler, with linker and conditional assembly which will run under OS9/68000.

Do any of your readers know of such a product?

Yours faithfully,
LEENSHIRE LTD.,

D.L.H. Smith

D.L.H. SMITH
Senior Project Engineer

OS-9 SOFTWARE SOURCEBOOK

Microware has published the OS-9 SOFTWARE SOURCEBOOK to provide OS-9 Users with a list of available Application Software. The list was compiled from information supplied by suppliers, as well as from catalogs and advertisements. They point out that being listed in the SOURCEBOOK in no way constitutes an endorsement of the product by Microware, nor are they responsible for the accuracy of completeness of the information or the performance, availability, or completeness of the programs listed.

The 37 page SOURCEBOOK begins with some info on the OS-9 Users Group (this page was blank in our book), the Compuserve OS-9 SIG, and the Unix Network USENET. It then breaks the approximately 75 Software Listings into sections on Business and Productivity; Test Editors and Word Processors; Database Systems; Communications Software; Assemblers, Cross-Assemblers, Simulators, and Translators; and Utility Programs. Each listing includes the Application, the Supplier, Software Requirements, Hardware Requirements, and a Description of the Software Product. A SOFTWARE SUPPLIER ADDRESS listing and SOFTWARE DISTRIBUTOR listing is provided at the end of the SOURCEBOOK.

A directory such as this is hard to maintain in such a fluid industry, and such things as errors and omissions, listings for products that are no longer available, etc., are bound to happen. The Publication of the SOURCEBOOK should generate feedback from Suppliers and Users alike that will help make future revisions more accurate. In an effort to help in keeping everyone up-to-date in this area, if you will send us an info copy of any corrections that you send to Microware, we will publish the letter in the Bit Bucket section of 68' Micro Journal.

Those corrections that are "Near to our Heart" include;

1. The O-F File Transfer Utility listed on page 28 shows the Supplier to be Datacomp, not Southeast Media. That is close enough to home that it would not cause any problems, but Datacomp is not listed in the SOFTWARE SUPPLIER ADDRESS section of the SOURCEBOOK (but Southeast Media IS).
2. The Southeast Media OS-9 Programs Basic09 XRef and OS-9 VDisk are not listed.
3. Lloyd I/O has appointed Southeast Media as their sole Marketing Outlet, including the establishment and support of Dealer, Distributor, and OEM Licensing Programs. Evidently, K-BASIC (which allows the easy transportation of a lot of FLEX-Based Software to OS-9, as well as providing a very flexible, Native Code BASIC Compiler for developing Application Software) was not available in time to make this issue of the SOURCEBOOK. This also affects the listings for CRASMB, CRASMB 16.32, OSM, and DO in the SOURCEBOOK.

In spite of the problems, I think any OS-9 User will find the OS-9 SOFTWARE SOURCEBOOK to be useful, and anticipate that further updates will be both more accurate and more complete as Software Suppliers realize the potential value of the SOURCEBOOK.

Dear Don,

I thought it might be a good idea to pass along to your readers, my thoughts and comments on a SUPER hard disk system that I have been using for software development.

The system is a 20 mega-byte unit from PERIPHERAL TECHNOLOGY in Marietta Georgia. I ordered this unit with the 20 Meg capacity early in the year when this size of drive was easy to come by. I now understand that it is very difficult to obtain this drive because the IBM systems are soaking them up. 5 and 10 meg sizes are much more plentiful and can be obtained easily from PT.

As for the system itself, I can't say enough about it's performance and the company's owner, Fredrick Brown.

Let me hit the hardware description first! The PT-69W came in a case that looks the same as big blues's PC. It has the same style, is 19 1/2 x 15 x 6 and has a slightly sloped front panel. My unit has the hard disk in the left side and two 40 track DSDD floppies on the right, one over the other in the convential manner. There are two serial ports on the back along with a centronics parallel port. All connectors are high quality DB-25s. The parallel port is connected in such a way that an IDC type of DB-25 connector can be pressed on one end of the cable and a Centronics compatable IDC connector on the other. This makes the connection very quick and simple! Inside, the unit is very

impressive. A Western Digital WD-1002 controller is used to interface to the hard disk. The parallel port buffer, cabling and all interconnections are neatly arranged and well placed. The only noticable difference that one sees from the standard FT-69 board, is a piggy-back daughter board used to interface to the WD-1002. The power supply which usually tends to be ignored, has not been ignored here; it is a high quality "switcher". This supply has plenty of power for all of the drives and the CPU. It appears to be a well built unit.

As for the operation of the unit and the software: I got STAR-DOS with my system and I am very impressed with Peter Stark's fine DOS. The utilities make it very comfortable to use. (Finally we have a DOS that is supported!) The real genius in this system is the disk drivers that Fredrick has written. They are absolutely the best! He has written them so that they have "descriptors" in them which allow mixing of any combination of drive sizes and types. They are very efficient and fast. They even allow the system to be booted from the hard disk and then, using the GET command to get the descriptor, you can change to the floppy as drive 0 ON THE FLY!!! These drivers are available for STAR-DOS when purchased with the system or can be purchased seperately if you are using FLEX. OS9 can also be purchased for the system.

Without question, this is not only the best system on the market, but it is the best value for dollar that I have found. My hat is off to PERIPHERAL TECHNOLOGY.

Jim Gerwitz
7907 E Wood Dr
Scottsdale AZ 85260

Marc I. Leavey, M.D.
6 Jenny Lane
Pikesville, Maryland
21208

May 4, 1985

Don Williams, Sr.
Publisher
68' Micro Journal
3900 Cassandra Smith Road
Hixson, TN 37343

Dear Don:

I have not had occasion to write you for some time. I believe since your heart surgery. I do hope, both from a professional and personal point of view, that it went well; and that you are once again chugging along without problems.

This letter kind of bubbled up to the surface after I received the June issue of the Journal. It has a few random thoughts, and I hope you enjoy them.

Did you see the April, 1985, issue of the Computer Shopper? On page 12 is an editorial entitled "SWTPC: Another Page For the History Book." In it, Stan Veit, the Editor-in-Chief of the magazine, gives a somewhat accurate, somewhat inaccurate view of the history of the SS-50 bus based machines. He omits a few little things, like SSB, and your magazine, and makes the system sound like it died before the invention of the sixteenth bit. Anyway, it makes for some novel reading, and I thought you might "enjoy" it. If you haven't seen it, let me know and I will photocopy it and send it down there.

Speaking of abberations, the page numbering in my issue runs from one through sixteen, then 25 through 40, then, 17 through 24, then 41 through the end, at page 64. If you had intended the June issue to be an Adventure Game, it would have been nice to warn us.

Speaking of ramblings, in your editorial, of the same name, in this issue, you ran down a list of early 6800 authors. I was, shall we say, taken aback, to see "Dr. Levy" after the well known names of Pete Stark and Mickey Ferguson. Is that me? I grant that it is spelled wrong, but I don't expect you to carry around the correct (screwy) spelling of my name in your head. If it's not me, thanks for the charge, anyway. I did publish in those days, usually in 73 and KiloBaud/Microcomputing.

I guess the only reason that you have not had anything submitted from me lately is that I am still playing with the obsolete 6800 running under the poorly supported BBB DOS. Trouble is, I run a medical office finance package in that system, which it does quite well, so I have little reason to change. Many of the utilities I write, such as Xmodem transfers, or buffer terminals, or sorts, or the like, are for that system which just is not within the interest realm of most of your readers. I have been bitten by the CoCo bug, however, and as soon as I get a tad more comfortable may well send something your way on that. I actually did write something already, routines to transfer the educational programs put out by the Children's Computer Workshop, and marketed by Tandy, on disk. You see, these programs, such as Ernie's Magic Shapes, and others, come on tape, and are written so that they cannot be directly put onto a disk system. After my kids got tired of waiting for the tape to load, I spent some time and converted them. Even though reviewers at both Hot CoCo and Rainbow commented in their reviews that the major problem with these programs is that they do not come, or could not be transferred to, disk, both of these magazines have rejected the article and listings on how to do it. If you're interested, let me know.

Last thing. I am running two of the DRC 64k static RAM boards in two 6800 systems. Did you know that they can be configured to hop around the 8000 I/O block, demanded by earlier 6800 systems, so that one board can serve as the whole memory? Would you be interested in an article on that?

I guess I've bent your ear, or eye, enough. Just had this accumulated pile of comments to send your way. I would have sent it on CompuServe, but don't know if you are there. I check into the CoCo SIG every couple of days, if you want to send Email, I am user code 75036,2501. If you would like this letter, or an edited version of it, for publication, let me know. I will chop out what you like and format it for the skinny columns.

My best to you and the gang in Tennessee. Keep on growing, but never forget your roots, for without them, you cannot blossom.

Sincerely,

Marc I. Leavey, M.D.

Ed's Note: Well Marc, I knew I would do it, just had to, what at 4:00 or so a.m., sorry it was you. I remember shaking my head (even thought about sticking it under running cold water, but the hour wasn't right) and saying to myself, "your going to blow this", and I did, again sorry! However, I am certain that all your readers knew who I was referring to - I sure did.

As to the articles, I hear more and more of the same, seems that they are devoting less and less time to article procurement. Guess they have their bios full, or something. I and thousands of other would appreciate your input, we are a SHARING group. And as to the 'history', see my letter** (edited) to them right after they published Stans history(?) - never did hear back from Stan, oh well.

As to the page numbers wandering all around, everyone around here knows better than to do it any other way than I do. Well, not exactly, actually it happens every now and then as the folding machine is and fed bundles of sheets at time. sometimes they do get jumbled. Could have told you it was put together the first of April (which it was), but what the heck, we may be goo but we're still human.

And as to our roots, well arc, we know who held us up on the way here. I deeply appreciate each and every one, for without that support and sharing, you would not be reading this today. We didn't make that big splash, but we didn't sink either. Kinda nice to think about and remember.

DMW

March 20, 1985

Computer Shopper

Attn: Mr. Stan Veit, Editor-in-Chief

Dear Stan,

What a surprise to read your recent editorial and/or hietorical review of the S-50 bus and SWTPC. It was nice to know that you are still around and kicking, as we have not heard from you for some years. Guess I, like you, don't read all the computer rags. As I am told you have been on your present job for quite some time now. Anyway, it was nice to see your article. However, I feel that you were sorte misleading, in some aspects.

As you should know we, S50 bus and 68XX(X are still VERY much alive and active! Actually the thing has not gone 'mainly' to industrial and control computing! Fact are - for the user group we direct our publications towards the breakdown is somewhat as follows - as of January 1985, our lateet audit period.

Domeatic (USA) - 47% of distribution
Overseas (51 foreign countries) - 53% distribution

As to S50 bus and 68XX(X) devices - well in Europe and the Far East - where we have reps - the 68XX(X) devices are either number ONE or TWO. Only here in the USA, where everyone but us seems to have the Big Blue syndrome, is the 8XXX aeries dominant. The entire industry here is going stale waiting for BB (Big Blue) to make the next move. Overseas, not so. That means that eventually they, because of more aggressive foreign development, not having to follow a pre-determined CPU and software protocol, will exceed what we have been doing here in the USA.

Just give the auto and other markets a thought, that have lagged behind while the foreign manufacturers kept on developing. Who would have thought, 15 years ago, that GM and Ford would have been so hard pressed by those 'silly, little Jap and Germany cars', not to mention TV/radio, camera, etc. Already the 68000 is fast becoming the 'standard' despite what BB does with it or other CPU devices.

When the shake-out comes - there will be some of us around who were there and on the S-50 bus. Maybe you did never look back, but some of us just kept looking forward. As a result, we are the oldest group of microcomputer users left, and SWTPC is the oldest microcomputer manufacturer in existence. Fact is, look in the current issue of 68 Micro Journal and see that they are still here, as well as GIMIX, SSB and many of the originals, or nearly so. We are not the largest, but we SURVIVE!! And that Stan, is the 'name of the game'. I would much rather have been small and survived, than big and failed, right?

Nice seeing you are still around, and hoping you might let your readers know that we still exist, all of us mostly.

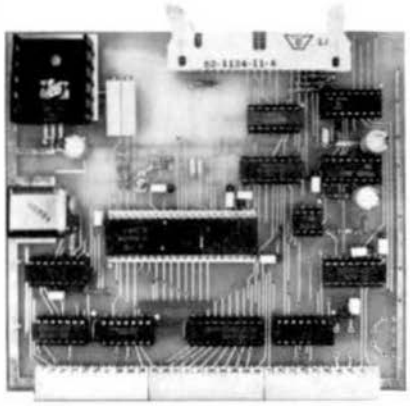
As to the historians ignoring the Dan Meyers and all the rest who were there, an more importantly, are STILL there, well it is always that way. Everyone seems to want to toot his own whistle, and ignore the others ringing in his ears.

With best personal regards,

Don Williams Sr.
Publisher

DMW/wp

OS-9 SUPPORT FOR FD-2



NEW! NEW!

Run double density on any S-50 6800 or 6809 computer. Who el e can offer this capability at these low prices? The FD-2 features:

- Control of up to four 5¼" OS/DD Drives
- SS-30 or SS-30C compatible
- Use Flex, OS-9, or Star Dos operating systems
- 2.0 MHZ operation with no "slow I/O" required
- Compatible with SWTPC DC1, DC2, DC3, or DC4 controllers

FD-2	Assembled/Tested ontroller Card	$149.95
DRV-68	6800 double density drivers + format program	$ 19.95
DRV-69	6809 double density drivers + format program	$ 29.95
DRV-09	FD-2 Disk Drivers for OS-9 (Source)	$100.00
STAR-DOS	For SWTPC & FD-2	$ 75.00

PERIPHERAL TECHNOLOGY

1480 Terrell Mill Rd., Suite 870

Marietta, Georgia 30067

VISA/MASTERCARD/CHECK/COD 404/973-0042

*OS-9 is a trademark of Microware and Motorola. Telex #880584

SOFTWARE DEVELOPERS!

YOU'VE JUST BEEN GIVEN THE BEST REASON YET TO GET OUR 68000/UNIX® DEVELOPMENT SYSTEM

THE VAR/68K® SERIES

VK-5XW20 (List price ~~$10,000~~) **$5,000.**
Includes: Terminal, 20 Mb hard disk, 512K RAM, 8 ports and REGULUS®

VK-5XW20T20 (List price ~~$12,900~~) **$6,500.**
Includes: all of above, plus 20 Mb tape streamer

RESELLERS!

Even more attractive specials are available to qualified resellers!

Smoke Signal has been designing, developing and manufacturing microcomputers based on the Motorola family of processors for the past six years. The VAR/68K is the most recent addition to our family of multi user computers.

VAR/68K is a registered trademark of Smoke Signal
REGULUS is a registered trademark of Alcyon Corp.
UNIX is a registered trademark of Bell Laboratories

Due to the extremely low prices being offered, we can only accept cash or C.O.D. orders, and we must limit purchases to one per customer. This is a limited time offer. Offer expires July 31, 1985.

TO OBTAIN YOUR VAR/68K AT THESE LOW PRICES, CONTACT:

SMOKE SIGNAL

CHAN VR COLINAS

WESTLAKE VILLAGE, CA 91302

(818) 889-8340 / Telex 910-494-4905

45 COOK ROAD,
NEWLANDS,
CAPE TOWN, 7700
SOUTH AFRICA.

14TH MAY 1985

DEAR DON,

RE: LOG.CMD 68MJ MAY 85.

PLEASE DISREGARD MY LETTER DATED 13TH APRIL 1985. TWO BUGS HAVE COME TO MY ATTENTION. THE VALIDITY OF THE FILESPEC IS NOT CHECKED. THIS CAN BE EASILY CORRECTED BY INSERTING 'LBCS BAD' AFTER THE LINE 'JSR GETFIL'.

BOB JONES OF CANADA HAS POINTED OUT AN ERROR IN THE 'LOGENO' SECTION OF THE CODE. IF FMS DETECTS AN ERROR IT GOES TO 'RPTERR' AND THEN BACK TO FLEX WITHOUT RESTORING THE VARIOUS VECTORS, MEMEND, ETC. THE LINE 'JMP WARMS' FOLLOWING THE LINE 'JSR RPTERR' SHOULD BE DELETED TO CORRECT THIS PROBLEM.

RE: COMMENTS ON DO.CMD
ALL REFERENCES TO THE Y REGISTER SHOULD HAVE READ U REGISTER. ALSO SAVING THE U REGISTER IS NOT SUFFICIENT. IT SHOULD POSSIBLY BE SAVED IN A TEMPORARY LOCATION IN THE RELOCATED PORTION OF THE DO COMMAND. THE INPUT REDIRECTION ROUTINE COULD THEN GET THE VALUE OF THE U REGISTER IN AN ABSOLUTE FASHION.

```
IE. STU UUUU    SAVE U REGISTER
    JSR DOCMND PROCESS THIS FILE LINE
    LOU UUUU    RESTORE U REGISTER
      .
      .
SVMEND RMB 2 SAVED MEMEND VALUE
UUUU RMB 2 TEMP FOR U REGISTER
      .
      .
PARAM EQU *
    PSHS B
    LOU UUUU  GET U REGISTER
      .
```

I HAVE FOUND TWO OTHER MINOR PROBLEMS. THE DO COMMAND LIKE THE OLD LOG COMMAND DOES NOT CHECK TO SEE IF MEMEND HAS BEEN ALTERED BY ANOTHER UTILITY. THE DO COMMAND DOES NOT WORK IN THE FLEX STARTUP FILE.

YOURS FAITHFULLY,

JOHN RITCHIE.

==========

Classified Advertising

TELETYPE Model 43 PRINTER - with serial (RS232) interface, and full ASCII keyboard. LIKE NEW - New cost $1295.00 - ONLY $559.00 ready to run - Call Tom - Larry - Bob, CPI 615 842-4600

***

S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09 CPU Card $1990. 1-DMAF2 Dual 8" Drives with Controller $2190. 1-CDS1 20 Meg Hard Disk System with Controller $2400.

1-S+ System with 256K RAM, 1-MPS4, 1-Parallel Port, MPU Processor Board $2700. 1-QW1 10 Meg 5" Hard Disk & 8" Disk Drive with DMAF3 Controller Board $3600. 1-X12 Terminal 12" $1260. 1-Cabinet for S+ System w/Filler Plate $500.

2-8212 Terminals $495.

Call Tom (615) 842-4600 M-F 9 A.M. to 5 P.M. E.S.T.

INDUSTRIAL PASCAL FOR THE 68000

If you're looking for a language to write real-time process control software, look no further. With the rising cost of labor, it is becoming critical that a high level language be used whenever possible. Find out why over 1400 companies have switched to OmegaSoft Pascal for their demanding applications.

OmegaSoft Pascal takes the Pascal framework and expands the basic data types, operators, functions, and memory allocation to fit the needs of real-time systems. These additions fit in the same structure as Pascal and enhance its usefulness without impairing the excellent readability, ease of maintenance, and structured design.

The compiler generates assembly language for assembly and link to run on the target system. Since a true relocating assembler and linking loader is used, only those runtime modules required are automatically linked in, providing a smaller object module than other compilers.

Large Pascal programs can be split up into conveniently sized modules to speed the development process. Procedures, functions, and variables can be referenced between Pascal modules and assembly language modules by using Pascal directives.

The compiler package includes an interactive, symbolic debugger. The debugger allows setting of breakpoints, displaying and changing variables, and tracing statements. Debugging can also be done at the assembly language level when needed. The debugger allows very fast turnaround for programs to be run on the host system (target system debugger coming soon).

The compiler package also includes a full relocatable macro assembler and linking loader. These are designed to support the compiler but may also be used for general assembly language development. In addition, a full screen editor is included which can be used with a variety of intelligent terminals.

Full source code is included for the runtime library, the debugger, the screen editor, and other support utilities.

Versions to run under the OS-9/68000 and VERSAdos operating systems are currently available to end-users and OEM's. End user price is $900 (domestic) or $925 (international). A version for CP/M-68K is available for OEM use, with OEM versions for UNIX type operating systems to follow.

Similar products to run on a 6809 system and generate 6809 code are also available for most major 6809 operating systems.

T.M. OmegaSoft is a trademark of Certified Software Corporation. OS-9/68000 is a trademark of Microware. VERSAdos is a trademark of Motorola. CP/M-68K is a trademark of DRI. UNIX is a trademark of Bell Labs.

CERTIFIED SOFTWARE CORPORATION

616 Camino Caballo. Nipomo. CA 93444
Telephone: (805) 349-0202: Telex: 467013

GOOD NEWS!

C for the 6809 WAS NEVER BETTER!

INTROL-C/6809,Version 1.5

Introl's highly acclaimed 6809 C compilers and cross-compilers are now more powerful than ever!

We've incorporated a totally new 6809 Relocating Assembler, Linker and Loader. Initializer support has been added, leaving only bitfield-type structure members and doubles lacking from a 100% full K&R implementation. The Runtime Library has been expanded and the Library Manager is even more versatile and convenient to use. Best of all, compiled code is just as compact and fast-executing as ever - and even a bit more so! A compatible macro assembler, as well as source for the full Runtime Library, are available as extra-cost options.

Resident compilers are available under **Uniflex, Flex** and **OS9.**

Cross-compilers are available for **PDP-11/UNIX** and **IBM PC/PC DOS** hosts.

Trademarks:
Introl-C, Introl Corporation
Flex and Uniflex, Technical Systems Consultants
OS9, Microware Systems
PDP-11, Digital Equipment Corp.
UNIX, Bell Laboratories
IBM PC, International Business Machines

For further information, please call or write.

INTROL
CORPORATION

647 W. Virginia St.
Milwaukee, WI 53204
(414) 276-2937

COMPILER EVALUATION SERVICES
By: Ron Anderson

The S.E. MEDIA Division of Computer Publishibg Inc., is offering the following SUBSCRIBER SERVICE:

COMPILER COMPARISON AND EVALUATION REPORT

Due to the constant and rapid updating and enhancement of numerous compilers, and the different utility, appeal, speed, level of communication, memory usage, etc., of different compilers, the following services are now being offered with periodic updates.

This service, with updates, will allow you who are wary or confused by the various claims of compiler vendors, an opportunity to review comparisons, comments, benchmarks, etc., concerning the many different compilers on the market, for the 6809 microcomputer. Thus the savings could far offset the small cost of this service.

Many have purchased compilers and then discovered that the particular compiler purchased either is not the most efficient for their purposes or does not contain features necessary for their application. Thus the added expense of purchasing additional compiler(s) or not being able to fully utilize the advantages of high level language compilers becomes too expensive.

The following COMPILERS are reviewed initially, more will be reviewed, compared and benchmarked as they become available to the author:

PASCAL "C" GSPL WHIMSICAL PL/9

Initial Subscription - $ 39.95
(includes 1 year updates)
Updates for 1 year - $ 14.50

S.E. MEDIA - C.P.I.
5900 Cassandra Smith Rd.
Hixson, Tn. 37343
(615) 842-4601

OS-9™ SOFTWARE

SDISK—Standard disk driver module allows the use of 35, 40, or 80 track double sided drives with COCO OS-9 plus you can read/write/format the OS-9 formats used by other OS-9 systems. **$29.95**

SDISK + BOOTFIX—As above plus boot directly from a double sided diskette **$35.95**

FILTER KIT #1—Eleven OS-9 utilities for "wild card" directory lists, copies, moves, deletes, sorts, etc. Now includes disk sector edit utility also. **$29.95** ($31.95)

FILTER KIT #2—Macgen command macro generator builds new commands by combining old ones with parameter substitution, 10 other utilities. **$29.95** ($31.95)

HACKER'S KIT #1—Disassembler and related utilities allow disassembly from memory, file. **$24.95** ($26.95)

PC-XFER UTILITIES—Utilities to read/write and format MS-DOS™ diskettes on CoCo under OS-9. Also transfer files between RS disk basic and OS-9 (requires sdisk). **$45.00**

BOLD prices are CoCo OS-9 format disk, other formats (in parenthesis) specify format and OS-9 level. All orders prepaid or COD, VISA and MasterCard accepted. Add $1.50 S&H on prepaid, COD actual charges added.

SS-50C
1 MEGABYTE RAM BOARD

Full megabyte of ram with disable options to suit any SS-50 6809 system. High reliability, can replace static ram for a fraction of the cost. $895 for 2 Mhz or $995 for 2.25 Mhz board assembled, tested and fully populated. (Add $6 shipping and insurance, quantity discounts available.)

D.P. Johnson, 7655 S.W. Cedarcrest St.
Portland, OR 97223 (503) 244-8152
(For best service call between 9-11 AM Pacific Time.)

OS-9 is a trademark of Microware and Motorola Inc.
MS-DOS is a trademark of Microsoft, Inc.

New From ELEKTRA™

"Modern styled cabinet for dual 5-1/4" Winchester drives or four half height floppies. 5v @ 5a. 12v @ 5a (8a peak). EMI filter. Fan $250.00
"DMA controller for dual 5-1/4" Winchesters and dual 5-1/4" floppies (four drives total) $695.00
FLEX™ or OS-9™ drivers 50.00
Complete ELEKTRA™ product line in stock
Upgrade your old systems
68008 available shortly for the SS-50 bus
With ELEKTRA™, you have expandability with quality
Software from Technical Systems Consultants and Microware Systems Corp.

Write or phone for current pricing

AAA Chicago Computer Center
120 Chestnut Lane — Wheeling, IL 60090
(312) 459-0450
Technical Consultation available most weekdays from 4 PM to 6 PM CST

Bulletin Board Software

Auto Answer — Mail
Ordertaking — Maintenance Programs
Used by large corporations for internal correspondence. 3-1/2 years in development by Paris Radio Electronics.
UniFLEX® version written in Extended Basic with source $395.00
OS-9™ version written in BasicO9™ with source 395.00
OS-9™ version for Coco (Special) 195.00

Other Communication Software

OS-9™ Super Modem Program by Epstein Associates $100.00
Super Modem Program for FLEX™ or SSB by AAACCC 75.00

ELEKTRA is a trademark of AAA Chicago Computer Center
FLEX is a trademark of Technical Systems Consultants, Inc.
OS-9 and BasicO9 are trademarks of Microware Systems Corp.
UniFLEX is a registered trademark of Technical Systems Consultants, Inc.

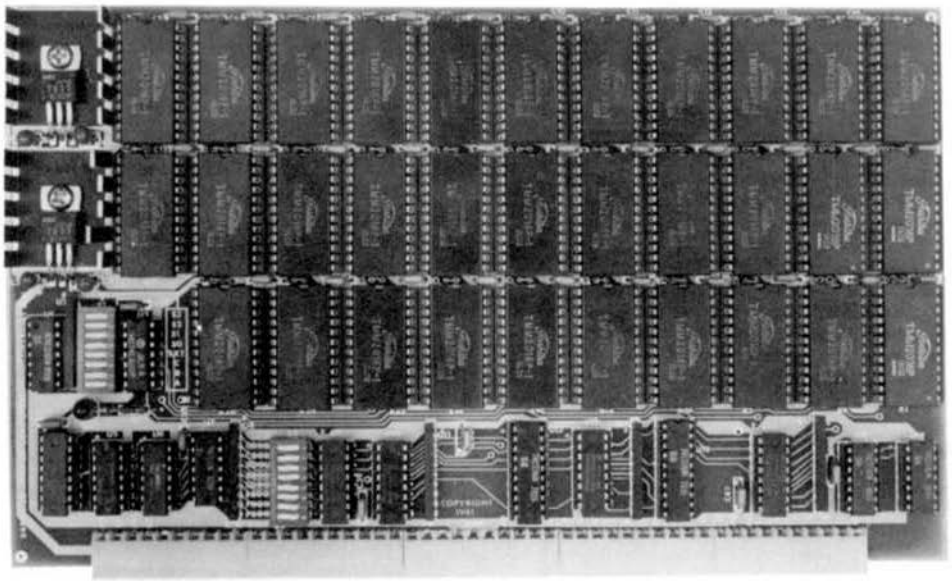
64K SS-50 STATIC RAM

PRICE CUT!!

$119.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $45

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$129
64K	$139

CLOSE OUT SPECIAL

WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

SOFTWARE FOR 680x SYSTEMS

SUPER SLEUTH DISASSEMBLERS
EACH $99-FLEX $101-OS/9 $100-UNIFLEX
OBJECT-ONLY versions: EACH $50-FLEX,OS/9,COCO
interactively generate source on disk with labels. include xref, binary editing
specify 6800,1,2,3,5,8,9/6502 version or Z80/6808.5 version
OS/9 version also processes FLEX format object file under OS/9
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/6808.5) only

CROSS-ASSEMBLERS (TRUE ASSEMBLERS, NOT MACRO SETS)
EACH $50-FLEX,OS/9,UNIFLEX ANY 3 $100 ALL $200
specify for 180x,6502,6801,6804,6805,6809,Z8,Z80,6048,8051,8085,68000
true, modular, free-standing cross-assemblers in C. with load/offset utilities
8-bit (not 68000) sources included with all cross-assemblers (for $200)

DEBUGGING SIMULATORS FOR POPULAR MICROPROCESSORS
EACH $75-FLEX $100-OS/9 $80-UNIFLEX
OBJECT-ONLY versions: EACH $50-COCO FLEX,COCO OS/9
interactively simulate processors, include disassembly formatting, binary editing
specify for 6800/1, (1/4)6805, 6502, 6809 OS/9, Z80 FLEX

ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809
6502 to 6809 $75-FLEX $85-OS/9 $80-UNIFLEX
6800/1 to 6809 & 6809 to position-ind. $50-FLEX $75-OS/9 $50-UNIFLEX

FULL-SCREEN XBASIC PROGRAMS with cursor control
AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS

DISPLAY GENERATOR/DOCUMENTOR	$50 w/source, $25 without
MAILING LIST SYSTEM	$100 w/source, $50 without
INVENTORY WITH MRP	$100 w/source, $50 without
TABULA RASA SPREADSHEET	$100 w/source, $50 without

DISK AND XBASIC UTILITY PROGRAM LIBRARY
$50-FLEX $30-UNIFLEX/MSDOS
edit disk sectors, sort directory, maintain master catalog, do disk sorts,
resequence some or all of BASIC program, xref BASIC program, etc.
non-FLEX versions include sort and resequence only

CMODEM TELECOMMUNICATIONS PROGRAM
$100-FLEX,OS/9,UNIFLEX
OBJECT-ONLY versions: EACH $50-FLEX,OS/9
menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc.
for COCO and non-COCO; drives internal COCO modem; port up to 2400 Baud

HARDWARE & SERVICES

5.25" DISKETTES
EACH 10-PACK $12.50-SSSD/SSDD/DSDD $20-DSOD
Amaray-Media, guaranteed 100% quality, with Tyvek jackets, hub rings, and labels

SS-50C 256K 1.5MHz MEMORY BOARDS
EACH BLANK $80 ASSEMBLED AND TESTED $350
with instruction manual, schematics, and delay line; all parts readily available

ADDITIONAL SERVICES FOR THE COMPUTING COMMUNITY
CUSTOMIZED PROGRAMMING
we will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications
CONTRACT PROGRAMMING
we will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 68000, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task
CONSULTING
we offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone 404-483-1717 or 4570

We take orders at any time, but please plan discussions after 5, if possible.

Contact us about catalog, dealer, discounts, and services.
Most programs in source: give computer, OS, disk size.
25% off multiple purchases of same program on one order.
VISA and MASTER CARD accepted; US funds only, please.
Add GA sales tax (if in GA) and 5% shipping.
(UNIFLEX tm Technical Systems Consultants; OS/9 tm Microware; COCO tm Tandy)

SOFTWARE FOR THE HARDCORE

** FORTH PROGRAMMING TOOLS from the 68XX&X **
** FORTH specialists — get the best!! **

NOW AVAILABLE — A variety of rom and disk FORTH systems to run on and/or do TARGET COMPILATION for

6800, 6301/6801, 6809, 68000, 8080, Z80

Write or call for information on a special system to fit your requirement.

Standard systems available for these hardware—

EPSON HX-20 rom system and target compiler
6809 rom systems for SS-50, EXORCISER, STD, ETC.
COLOR COMPUTER
6800/6809 FLEX or EXORCISER disk systems.
68000 rom based systems
68000 CP/M-68K disk systems, MODEL II/12/16

tFORTH is a refined version of FORTH Interest Group standard FORTH, faster than FIG-FORTH. FORTH is both a compiler and an interpreter. It executes orders of magnitudes faster than interpretive BASIC. MORE IMPORTANT, CODE DEVELOPMENT AND TESTING is much, much faster than compiled languages such as PASCAL and C. If Software DEVELOPMENT COSTS are an important concern for you, you need FORTH!

firmFORTH™ is for the programmer who needs to squeeze the most into roms. It is a professional programmer's tool for compact rommable code for controller applications.

™ tFORTH and firmFORTH are trademarks of Talbot Microsystems
™ FLEX is a trademark of Technical Systems Consultants, Inc.
™ CP/M-68K is trademark of Digital Research, Inc.

tFORTH™ from TALBOT MICROSYSTEMS NEW SYSTEMS FOR 6301/6801, 6809, and 68000

---> tFORTH SYSTEMS <---

For all FLEX systems: GIMIX, SWTP, SSB, or EXORcisor Specify 5 or 8 inch diskette, hardware type, and 6800 or 6809.

** tFORTH — extended fig FORTH (1 disk) $100 ($15)
with fig line editor.
** tFORTH + — more! (3 5" or 2 8" disks) $250 ($25)
adds screen editor, assembler, extended date types, utilities, games, and debugging aids.
** TRS-80 COLORFORTH — available from The Micro Works
** firm FORTH — 6809 only. $350 ($10)
For target compilations to rommable code.
Automatically deletes unused code. Includes HOST system source and target nucleus source. No royalty on targets. Requires but does not include tFORTH +.
** FORTH PROGRAMMING AIDS — elaborate decompiler $150

** tFORTH for HX-20, in 16K roms for expansion unit or replace BASIC $170

** tFORTH 68K for CP/M-68K 8" disk system $290
Makes Model 16 a super software development system.

** Nautilus Systems Cross Compiler
— Requires: tFORTH + HOST + at least one TARGET:
— HOST system code (6809 or 68000) $200
— TARGET source code: 6800-$200, 6301/6801—$200
same plus HX-20 extensions— $300
6809—$300, 8080/Z80—$200, 68000—$350

Manuals available separately — price in ( ).
Add $6 system for shipping, $15 for foreign air.

TALBOT MICROSYSTEMS 1927 Curtis Ave., Redondo Beach, CA 90278 (213) 376 9941

WINDRUSH MICRO SYSTEMS

UPROM II

PROGRAMS and VERIFIES: I2758, I2508, I2716, I2516, I2732/2732B, MCM6876/6, I2764/2764A, I2564, I27128/27128A, and I27256. I=Intel, T=Texas, M=Motorola.

NO PERSONALITY MODULES REQUIRED!

TRI-VOLT EPROMS ARE NOT SUPPORTED

INTEL's inteligent programming (tm) implemented for Intel 2764, 27128 and 27256 devices. Intelligent programming reduces the average programming time of a 2764 from 7 minutes to 1 minute 15 seconds (under FLEX) with greatly improved reliability.

Fully enclosed pod with 5' of flat ribbon cable for connection to the host computer MC6821 PIA interface board.

MC6809 software for FLEX and OS9 (Level 1 or 2, Version 1.2).

BINARY DISK FILE offset loader supplied with FLEX, MDOS and OS9.

Menu driven software provides the following facilities:

a. FILL ..... a selected area of the buffer with a HEX char.
b. MOVE ..... blocks of data.
c. DUMP ..... the buffer in HEX and ASCII.
d. FIND ..... a string of bytes in the buffer.
e. EXAMINE/CHANGE . the contents of the buffer.
f. CRC ..... checksum a selected area of the buffer.
g. COPY ..... a selected area of an EPROM into the buffer.
h. VERIFY ..... a selected area of an EPROM against the buffer.
i. PROGRAM ..... a selected area of an EPROM with data in the buffer.
j. SELECT ..... a new EPROM type (return to types menu).
k. ENTER ..... the system monitor.
l. RETURN ..... to the operating system.
m. EXECUTE ..... any DOS utility (only in FLEX and OS9 versions).

FLEX AND OS9 VERSIONS AVAILABLE FROM GIMIX. SSB/MDOS CONTACT US DIRECT.

PL/9

- Friendly inter-active environment where you have INSTANT access to the Editor, the Compiler, and the Trace-debugger, which, amongst other things, can single step the program a SOURCE line at a time. You also have direct access to any FLEX utility and your system monitor.
- 375+ page manual organized as a tutorial with plenty of examples.
- Fast SINGLE PASS compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files.
- Signed and unsigned BYTES and INTEGERS, 32-bit floating point REALS.
- Vectors (single dimension arrays) and pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (\), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/XOR)
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (XREG), (CCR) and (STACK).
- FULLY supports the MC6809 RESET, NMI, FIRQ, IRQ, SWI, SWI2, and SWI3 vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute SNAP!
- Machine code may be embedded in the program via the 'GEN' statement. This enables you to code critical routines in assembly language and embed them in the PL/9 program (see 'MACE' for details).
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library procedure modules are supplied: IOSUBS, BITIO, HASBIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, BASTRING, and REALCON.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Anderson's FLEX User Notes column in '68. Need we say more?

MACE/XMACE/ASM05

All of these products feature a highly productive environment where the editor and the assembler reside in memory together. Gone are the days of tedius disk load and save operations while you are debugging your code.

- Friendly inter-active environment where you have instant access to the Editor and the Assembler, FLEX utilities and your system monitor.
- MACE can also produce ASMPROCs (GEN statements) for PL/9 with the assembly language source passed to the output as comments.
- XMACE is a cross assembler for the 6800/1/2/3/8 and supports the extended mnemonics of the 6303.
- ASM05 is a cross assembler for the 6805.

D-BUG

LOOKING for a single step tracer and mini in-line disassembler that is easy to use?? Look no further, you have found it. This package is ideal for those small assembly language program debugging sessions. D-BUG occupies less than 6K (including its scratch and variables) and may be loaded anywhere in memory. All you do is LOAD IT, AIM IT and GO! (80 col VDUs only).

McCOSH 'C'

This is as complete a 'C' compiler as you will find on any operating system for the 6809. It is completely compatible with UNIX V7! and only lacks 'bit-fields' (which are of little practical use in an 8-bit world!).

- Produces very efficent assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%.
- Supports interleaved assembly language programs.
- INCLUDES its own assembler. The TSC relocating assembler is only required if you want to generate your own libraries.
- The pre-processor, compiler, optimizer, assembler and loader all run independently or under the 'CC' executive. 'CC' makes compiling a program to executable object as simple as typing in 'CC,HELLO.C <RETURN>'.

IEEE-488

- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:
  - Talker - Serial Poll - Single or Dual Primary Address
  - Listener - Parallel Poll - Secondary Address
  - System Controller - Group Trigger - Talk only ... Listen only
- Fully documented with a complete reprint of the HILLHARD article on the IEEE bus and the Motorola publication 'Getting aboard the IEEE Bus'.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. A complete back to back test program is also supplied in the form of a listing. These drivers have been extensively tested and are GUARANTEED to work.
- Single S-30 board (4, 8 or 16 addresses per port), fully socketed, gold plated bus connectors and IEEE interface cable assembly.

PRICES

D-BUG	(6809 FLEX only)	$ 75.00
MACE	(6809 FLEX only)	$ 75.00
XMACE	(6809 FLEX only)	$ 98.00
ASM05	(6809 FLEX only)	$ 98.00
PL/9	(6809 FLEX only)	$198.00
'C'	(6809 FLEX only)	$295.00
IEEE-488	with IEEE-488 cable assembly	$298.00
UPROM-II/U	with one version of software (no cable or interface)	$395.00
UPROM-II/C	as above but complete with cable and S-30 interface	$565.00
CABLE	5' twist-n-flat 50 way cable with IDC connectors	$ 35.00
S-30 INT	S-30 interface for UPROM-II	$130.00
EXOR INT	Motorola EEQBus (EXORciser) interface for UPROM-II	$195.00
UPROM SFT	Software drivers for 2nd operating system. Specify FLEX or OS9 AND disk size!	$ 35.00
UPROM SRC	Assembly language source (contact us direct)	

ALL PRICES INCLUDE AIR MAIL POSTAGE

Terms: CWO. Payment by Int'l Money Order. VISA or MASTER-CARD also accepted.

WORSTEAD LABORATORIES, NORTH WALSHAM, NORFOLK, ENGLAND. NR28 9SA.

TEL: 44 (692) 404086
TLX: 975548 WMICRO G

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, LLOYD I/O, & ALFORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporation, MDOS (tm) and EXORciser (tm) are trademarks of Motorola Incorporated.

68' MICRO JOURNAL

The only All 6809, 68000 Computer Magazine !
★ More 6809, 68000 material
★ than all the others Combined !

MAGAZINE COMPARISON
(2 years)
Monthly Averages

	6800 Articles			TOTAL
KB	BYTE	CC	DOBB'S	PAGES
7.8	6.4	2.7	2.2	19.1 ea. mo.

Average cost for all four each month: $6.53
(Based on advertised 1-year subscription price)

'68' cost per month: $2.04

That's Right! Much, Much More

for About

1/3 the Cost!

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Master Charge ☐ — VISA ☐

Card # _____ Exp. Date _____

For ☐ 1-Year ☐ 2 Years ☐ 3 Years

Enclosed: $_____

Name _____

Street _____

City _____ State _____ Zip _____

My Computer Is: _____

Subscription Rates
(Effective March 3, 1985)

U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50

*** Foreign Surface: Add $12.00 per Year to USA Price.**

*** Foreign Airmail: Add $48.00 per Year to USA Price.**

*** Canada & Mexico: Add $ 9.50 per Year to USA Price.**

*** U.S. Currency Cash or Check Drawn on a USA Bank !!**

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

☎ **(615)842-4600**

TELEX 558 414 PVT BTH

VISA MasterCard

STAR-DOS LEVEL I

Whenever a new DOS is introduced, there's always the problem of developing software to work with it. So we did it the opposite way — we analyzed the requirements of software that already exists and developed a DOS that met them... and exceeded them! The result is STAR-DOS Level I, a new DOS for 6809 systems, ideal for single-user industrial, control, and advanced hobbyist applications. This includes SS-50 systems and single-board computers from a variety of vendors.

Level I is compatible with most current 6809 hardware and software. On the hardware side, it allows up to ten floppy or Winchester drives with appropriate controllers. On the software side, it runs existing 6809 software from all the major 6809 software suppliers, including TSC, Star-Kits, Introl, and others.

Write or call for more information. STAR-KITS Software Systems Corporation. P.O. Box 209, Mt. Kisco N.Y. 10549 (914) 241-0287.

STAR-KITS

ANDERSON COMPUTER CONSULTANTS
&
Associates

Ron Anderson, respected author and columnist for 68 MICRO JOURNAL announces the **Anderson Computer Consultants & Associates,** a consulting firm dealing primarily in 68XX(X) software design. Our wide experience in designing 6809 based control systems for machine tools is now available on a consultation basis.

Our experience includes programming machine control functions, signal analysis, multi-axis servo control (CNC) and general software design and development. We have extensive experience in instrumentation and analysis of specialized software. We support all popular languages pertaining to the 6809 and other 68XX(X) processors.

If you are a manufacturer of a control or measuring package that you believe could benefit from efficient software, write or call Ron Anderson. The fact that any calculation you can do with pencil and paper, can be done much better with a microcomputer. We will be happy to review your problem and offer a modern, state-of-the-art microcomputer solution. We can do the entire job or work with your software or hardware engineers.

Anderson Computer Consultants & Associates
3540 Sturbridge Court
Ann Arbor, MI 48105

COMPARE

our EPROM PROGRAMMER with the field.

All data taken directly from manufacturer's current advertising. Software, interfaces, or personality modules may also be required at additional cost.

- Triple voltage EPROM
- Supplied in kit form

	A	B	C	D	E	F
INTERFACE	S30	PAR	PAR	SER	S30	SER	SER
INTELLIGENT	NO	NO	NO	YES	NO	YES	YES
PROGRAMS							
2704*			•				•
2508	•			•	•	•	
2708*			•				•
2758	•	•	•	•		•	
2516	•	•	•	•	•	•	
2716	•	•	•	•			•
2716*			•		•		•
2532	•		•	•	•	•	•
2732	•		•	•	•	•	•
2732A	•		•	•	•	•	•
2564	•		•	•	•	•	•
2764	•		•	•	•	•	•
2528	•						
27128	•						
2816							•
68764			•				
8748						•	
8749						•	
TOTAL	11	3	12	6	11	11	11
PRICE	$125	$45*	$169	$289	$375	$489	$575

EPROM EPROM Programmer, $125. Personality module for 2508, 2758, 2516, and 2716 included. Specify CPU, disk size, and operating system (TSC's FLEX or 6809's OS9) when ordering. Manual only, $10; refundable with EPROM purchase.

UNITEK • P.O. Box 671 • Emporia, VA 23847

TMP SOFTWARE ANNOUNCES

****

A Popular FLEX Database Program, DATAMAN

Is being Re-released!

****

$195.00

POWERFUL CAPABILITIES:

- Each field can contain up to 126 characters, and each field name can contain up to 29 characters.
- Field types can be alpha, numeric or monetary.
- **DATAMAN** lets you produce vertical or horizontal reports of your database, or create customized letters and forms to pull data from a database using **DATAMAN** files processed by the TSC Text Processor.
- **DATAMAN** allows you to look up records quickly, to merge unlike databases and to move records from one database into another.
- **DATAMAN** includes a Label Generator to produce mailing, shipping or inventory labels.
- To sort records based on the geographic, historical, name, or monetary criteria you select, **DATAMAN** creates Sort Files which can then be used with the TSC Sort Merge package.

GIVE YOU MORE ABILITIES:

Our Users put **DATAMAN** to work for them to do customer mailings, past due notices, real estate listings, invoicing, sales analysis, activity scheduling, inventories, employee records and client profile reports.

Also included in this package is **DATARAND**, the extension program to **DATAMAN** which enhances **DATAMAN** programs.

DATAMAN runs under TSC Extended Basic. The TSC Sort Merge and Text Processor programs are also recommended. The Source Code for DATAMAN and DATARAND is included on the release disks.

ORDERING INFORMATION: TMP SOFTWARE
2431 E. Douglas • Wichita, KS • 67211
OR CALL TOLL FREE: 1-800-255-1382 Ext. 47
We accept VISA, MC, AMEX, money orders and checks.

NOTE: TMP Software also offers the TMP POWER MANAGER for OS-9 systems, and the TMP FREEFORM FILER for OS-9 and Unifies systems.
Flex and TSC are trademarks of Technical Systems Consultants, Inc.

68' MICRO JOURNAL

Disk- 1 Filesort, Minicat, Minicopy, Minifms, **Lifetime, **Poetry, **Foodlist, **Diet.

Disk- 2 Diskedit w/ inst.& fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpawn,**Lifetime

Disk- 3 Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave.

Disk- 4 Mailing Program, *Finddat, *Change, *Testdisk.

DISK- 5 *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.

Disk- 6 **Purchase Order, Index (Disk file indx)

Disk- 7 Linking Loader, Rload, Harkness

Disk- 8 Crtest, Lanpher (May 82)

Disk- 9 Datecopy, Diskfix9 (Aug 82)

Disk-10 Home Accounting (July 82)

Disk-11 Dissembler (June 84)

Disk-12 Modem68 (May 84)

Disk-13 *Initmf68, Testmf68, *Cleanup, *Dskalign, Help

Disk-14 *Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib

Disk-15 Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo)

Disk-16 Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc, Date.Txt

Disk-17 Match Utility, RATBAS (A Basic Preprocessor) June 85

NOTE:

This is a reader service **ONLY!** No Warranty is offered or implied, they are **as received** by **'68' Micro Journal,** and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other.

PRICE: 8" Disk **$14.95** - 5" Disk **$12.95**

68' Micro Journal
5900 Cassandra Smith Rd.
Hixson, Tn. 37343
(615)842-4600

* Indicates 6800
** Indicates BASIC SWTPC or TSC 6809 no Indicator.

MASTER CARD - VISA Accepted
Foreign -- add **10%** for Surface
or **20%** for Air!!

TRS-80+ MOD I, III, COCO, TI99/4a
TIMEX 1000, OSBORNE, others

GOLD PLUG - 80

Eliminate disk reboots and data loss due to oxidized contacts at the card edge connectors.
GOLD PLUG 80 solders to the board edge connector. Use your existing cables. (if gold plated)

GOLD PLUG 80 Mod I (6)	$44.95	~~$54.95~~
Keyboard/EI (mod I)	15.95	~~18.95~~
Individual connectors	7.95	~~9.95~~
COCO Disk Module (2)	16.95	~~18.95~~
Ground tab extensions	INCL	~~1.00~~
Disk Drives (all R.S.)	7.95	~~9.95~~
Gold Disk Cable 2 Drive		29.95
Four Drive Cable		39.95
GOLD PLUG 80 Mod III (6)		54.95
Internal 2 Drive Cable		29.95
Mod III Expansion port		10.95

SPECIAL PRICES

COCO MODULE INSTALLATION AVAILABLE

USA shipping $1.45 — Can/Mex $4.
Foreign $7. — TEXAS 5% TAX

Ask your favorite dealer or order direct

E.A.P. CO.
P.O. BOX 14
KELLER, TEXAS 76248
(817) 498-4242

ORDER TODAY! — MC/VISA

+ trademark Tandy Corp

NEW

XPNDR2™ for the CoCo DISK SYSTEM

XPNDR2 $39.95 each or **2/$76**
This prototype card features a 40 pin connector for projects requiring an on-line disk system or ROM paks. The CoCo signals are brought out to wire-wrap pins. Special gold plated spring clips provide reliable and noisefree disk operation plus solid support for vertical mounting of the controller. The entire 4.3 × 7 inch card is drilled for ICs. Assembled, tested and ready to run.

XPNDR1 $19.95 each or **2/$36**
A rugged 4.3 × 6.2 inch bare breadboard that brings the CoCo signals out to labeled pads. Both XPNDR cards are double-sided glass/epoxy, have gold plated edge connectors, thru-hole plating and are designed with heavy power and ground buses. They're drilled for standard 0.3 and 0.6 inch wide dual in-line wirewrap sockets, with a 0.1 inch grid on the outboard end for connectors.

SuperGuide $3.95 each
Here is a unique plastic insert that aligns and supports printed circuit cards in the CoCo cartridge port. Don't forget to **ORDER ONE FOR YOUR XPNDR CARDS**.

Included with each XPNDR card are 8 pages of APPLICATION NOTES to help you learn about chips and how to connect them to your CoCo.

To order or for technical information call:

(206) 782-6809
weekdays 8 a.m. to noon

We pay shipping on prepaid orders. For immediate shipment send check, money order or the number and expiration date of your VISA or MASTERCARD to:

ROBOTIC MICROSYSTEMS
BOX 30807 SEATTLE, WA 98103

LLOYD I/O™
Computer Engineers

19535 NE GUSAN • PORTLAND, OR 97230 (USA)
PHONE: (503) 666-1097 • TELEX: 910 380 5448 LLOYD I O

K-BASIC™ *IS HERE*

K-BASIC is a TSC XBASIC (XPC) compatible COMPILER
for OS9 & FLEX...price $199

Here at last is a compiler for BASIC that will compile all your XBASIC programs. K-BASIC compiles TSC's XBASIC and XPC programs to machine code. K-BASIC is ready now to save you money and time by teaching your computer to:
• Think Faster • Conserve Memory • Be Friendlier

Call (503) 666-1097 for our CATALOG.
We have many programs for serious software developers!

DO™
Micro BASIC for OS9...$149

A structured mirco BASIC for general system control featuring: Parameter passing, 10 string variables, 26 numeric variables, subroutines, nested loops, interactive I/O, sequential files, and time variables (for applications executing in the background required to execute procedures such as disk or file backups.) Includes the SEARCH and RESCUE UTILITIES™. (For OS9 ONLY.)

SEARCH and RESCUE UTILITIES™
for OS9...$35

A super directory search utility. Output may be piped to the included utilities to perform file COPIES, DELETES, MOVES, LISTING (pagination), and FILTERING. Some filtering utility programs are included; of interest is the FILE DATE CHECKING utilities YOUNGER and DRAFT (Level 2). (For OS9 Level 1 and 2.)

PATCH™
Modem Communications for OS9...$39

PATCH is a modem communications program for OS9 featuring: KEY MACROS, ASCII TEXT AND BINARY FILE UP/DOWN LOADING, PRINTER COPY, and HELP MENUS. We use it several times each day with our TELEX service. PATCH is convenient and easy to use. Key macros may be pre-stored and loaded at any time.

CRASMB™
CROSS ASSEMBLER PACKAGE
for OS9 & FLEX...all for $399
Motorola CPU's...$150
Intel CPU's...$150, Others...$150

CRASMB is the highly acclaimed cross assembler package for OS9 and FLEX systems. It turns your 6809 computer into a development station for these target CPUs:

6800 6801 6804 6805 6809 6811 6502
7000 1802 8048 8051 8080 8085 Z8 Z80

(68000 16/32 bit cross assembler...$249)

CRASMB features: Macros, Conditionals, Long symbol names, Symbol cross reference tables, Object code in 4 formats (OS9, FLEX, S-1-S9, INTEL HEX).

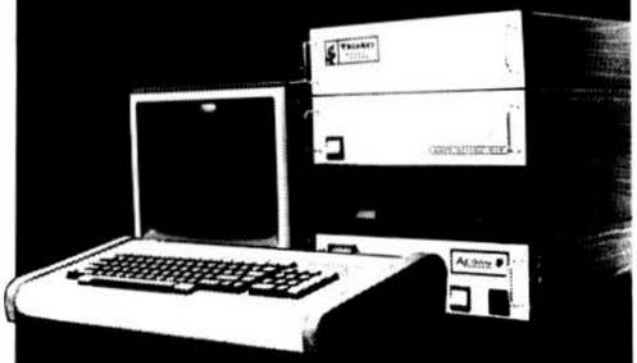
VISA, MC, COD, CHECKS, ACCEPTED

USA:	LLOYD I/O (503 666 1097),	S.E. MEDIA (800 338 6800)
England:	Vivaway (0582 423425),	Windrush (0692 405189)
Germany:	Zocher Computer (65 25 299),	Keil Software (06203 6741)
Australia:	Paris Radio Electronics (344 9111)	

K-BASIC, DO, SEARCH and RESCUE UTILITIES, PATCH, CRASMB and CRASMB 16/32 are trademarks of LLOYD I/O
OS9 is a ™ of Microware. FLEX is a ™ of TSC

ACORN

COMPUTER SYSTEMS SS-50C



MODULES - BARE CARDS - KITS - ASSEMBLED & TESTED

Stackable Modules		KIT	A&T
20 amp POWER SUPPLY w/fan w/Disk protect relay		350.00	400.00
DISK CABINET w/regs. & cables less DRIVES		200.00	250.00
MOTHER BOARD, 8 SS-50c, 8 SS-30c NMI button		225.00	325.00
Item	**Bare**	**KIT**	**A&T**
IT3 - INTERRUPT TIMER 1, 10, 100 per sec.	19.95	29.95	39.95
PB4 - INTELLIGENT PORT BUFFER Single board comput.	39.95	114.95	139.95
DPIA - Dual PIA parallel port, 4 buffered I/Os	24.95	69.95	89.95
XADR - Extended Addressing BAUD gen. PIA port	29.95	69.95	89.95
MB8 - MOTHER BOARD SS-50c w/BAUD gen.	64.95	149.95	199.95
P16S - 16SK PROM DISK 21, 2764 EPROMs	39.95	79.95	109.95
FD8S - Firmware de elopment 2, 8K blocks	39.95	84.95	114.95
XMPR - 2764 PROM burner adapt. for 2716 BURNER		19.95	------
CHERRY Keyb ard w/Cabinet 96 key capacitive		249.95	------
TAXAN 12". 16 Mhz MONITOR GREEN		------	149.95
AMBER		------	159.95
4 MODULE CABINET - unfinished		150.00	------
POWER SUPPLY w/disk protect		250.00	------

Color Computer

MONOLINK - 20 Mhz Monochrome video driver		15.00	20.00
CC3O PORT BOS w/power supply 5 SS-30, 2 Cart		169.95	199.95
POWER BOX 6 switched outlets transient suppression		29.95	39.95
RS-232 3-switched ports for above	ADD	+20.00	+25.00

Write for FREE Catalog

ADD $3.00 S&H PER ORDER

WIS. ADD 5% SALES TAX

11931 W. Bluemound Road

MILWAUKEE, WIS. 53226

(414) 257-0300

68' MICRO JOURNAL ADVERTISERS INDEX

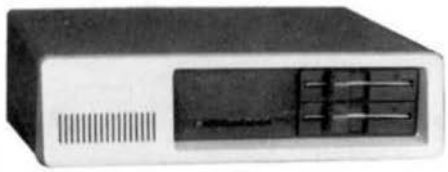
This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

PT-69 SINGLE BOARD COMPUTER SYSTEMS

NOW WITH WINCHESTER OR FLOPPY DISK DRIVES

The proven PT-69 Single Board Computer line is expanding! Systems now can be Winchester or floppy-based. Available also in a smaller cabinet without drives for dedicated systems with no mass storage requirements.

* 1 MHZ 6809E Processor
* Time-of-Day Clock
* 2 RS 232 Serial Ports (6850)
* 56K RAM 2K/4K EPROM
* 2 8-bit Parallel Ports (6821)
* 2797 Floppy Disk Controller



Winchester System

Floppy System

Custom Design Inquiries Welcome

- PT69XT WINCHESTER SYSTEM — $1895.95
  Includes 5 MEG Winchester Drive, 1 40-track DS/DD Drive, Parallel Printer Interface + choice of OS/9 or STAR-DOS
- PT69S2 FLOPPY SYSTEM — $ 949.95
  Includes PT69 Board, 2 DS/DD 40-TRK 5¼" drives, cabinet, switching power supply, OS-9 or STAR-DOS
- PT-69A ASSEMBLED & TESTED BOARD — $289
- OS/9 — $200
- STAR-DOS — $ 50

CALL OR WRITE FOR ADDITIONAL CONFIGURATIONS

PERIPHERAL TECHNOLOGY

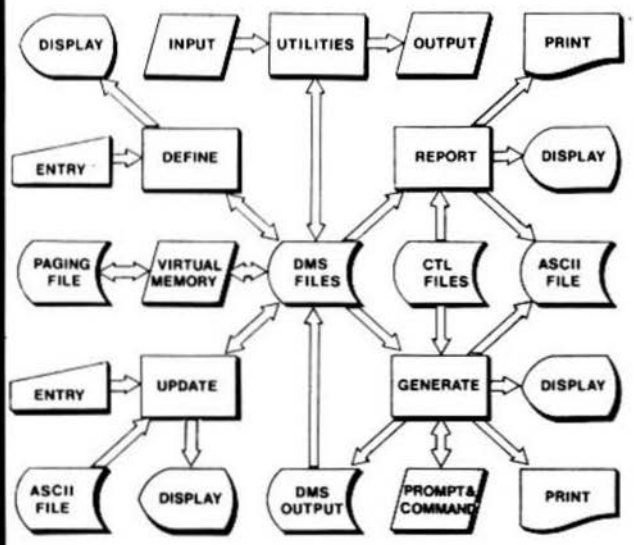
1480 Terrell Mill Rd., Suite 870
Marietta, Georgia 30067

VISA/MASTERCARD/CHECK/COD — Telex #880584 — 404/973-0042

[illegible] is a trademark of Motorola and Motorola

XDMS

Data Management System



System Architecture

WESTCHESTER Applied Business Systems
Post Office Box 187
Briarcliff Manor, N.Y. 10510

XDMS Data Management System

The XDMS Data Management System is available in three levels. Each level includes the XDMS nucleus, VMDEX utility and System Documentation for level III. XDMS is one of the most powerful systems available for 6809 computers and may be used for a wide variety of applications. XDMS users are registered in our database to permit distribution of product announcements and validation of user upgrades and maintenance requests.

XDMS Level I

XDMS Level I consists of DEFINE, UPDATE and REPORT facilities. This level is intended as an "entry level" system, and permits entry and reporting of data on a "tabular" basis. The REPORT facility supports record and field selection, field merge, sorting, line calculations, column totals and report titling. Control is via a English-like language which is upward compatible with level II. XDMS Level I $129.95

XDMS Level II

Level II adds to Level I the powerful GENERATE facility. This facility can be thought of as a general file processor which can produce reports, forms and form letters as well as file output which may be re-input to the facility. GENERATE may be used in complex processing applications and is controlled by a English-like command language which encompasses that used by Level I. XDMS Level II $199.95

XDMS Level III

Level III includes all of level II plus a set of useful DMS Utilities. These utilities are designed to aid in the development and maintenance of user applications and permit modification of XDMS system parameters, input and output of XDMS files, display and modification of file format, graphic display of numerical data and other functions. Level III is intended for advanced XDMS users. XDMS Level III $269.95
XDMS System Documentation only ($30, credit toward purchase). . $ 24.95

XACC Accounting System

The XACC General Accounting System is designed for small business environments of up to 10,000 accounts and inventory items. The system integrates accounting functions and inventory plus the general ledger, accounts receivable and payable functions normally sold separately in other systems. Features user defined accounts, products (or services), transactions, invoicing, etc. Easily configured to most environments. XACC General Accounting System (Requires XDMS, prof. Lv. III). . $269.95
XACC System Documentation only ($30, credit toward purchase). . $ 24.95

WESTCHESTER Applied Business Systems
Post Office Box 187, Briarcliff Manor, N.Y. 10510

All software is written in macro/assembler and runs under 6809 FLEX O/S. Terms: Check, Money Order, Visa or Mastercharge. Shipment first class. Add P&H $1.50 ($7.50 Foreign Surface or $15.00 Foreign Air). NY Res add sales tax. Specify 5" or 8".

Sales: S. E. [illegible], (615) 842-4600. Consultation: 914-941-3552 (evens)

GIMIX HAS THE 6809 SYSTEM TO SUIT YOUR NEEDS

HARDWARE

All systems feature the **GIMIX CLASSY CHASSIS**; with a ferro-resonant constant voltage power supply, gold plated bus connectors, and plenty of capacity for future expansion.

Static RAM and double-density DMA floppy disk controllers are used exclusively in all systems.

All systems are guaranteed for 2 MHz operation and include complete hardware and software documentation, necessary cables, filler plates, etc.

Systems are assembled using burned-in and tested boards, and all disk drives are tested and aligned by **GIMIX**.

You can add additional components to any system when ordering, or expand it in the future by adding **RAM, I/O,** etc.

GIMIX lets you choose from a wide variety of options to customize your system to your needs.

SOFTWARE

All **OS-9/FLEX** systems allow you to software select either operating system.

Also included is the **GMX8UG** monitor and, in systems with 128K or more of RAM, **GMX-VDISK** for **FLEX**.

All **GIMIX OS-9** systems include **Microware's Editor, Assembler, Debugger, Basic09,** and **Runb**; and the GMX versions of **RMS** and **DO** for **OS-9**.

All **GIMIX** versions of **OS-9** can read and write RS color computer format OS-9 disks, as well as the Microware/GIMIX standard format.

New and exclusive with **OS-9 GMX III** systems is the **GMX OS-9 Support ROM**, a monitor for **OS-9** that includes memory diagnostics and allows the system to boot directly from either hard disk or floppy.

A wide variety of languages and other software is available for use with either **OS-9** or **FLEX**.

OS-9 GMX III/FLEX SYSTEMS (#79)

The **#79** super system now includes (in addition to the above): the **GMX 6809 CPU III**, a **256K CMOS Static RAM Board (#72)**, and a **3-port Intelligent Serial I/O Processor (#11)**.

The **GMX 6809 CPU III** can perform high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

The 3-Port Intelligent Serial I/O Board (#11) significantly reduces system overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

with dual 40 track DSDD drives	**$5998.79**
with dual 80 track DSDD drives	**$6198.79**
with **#88** dual 8" DSDD drive system	**$7698.79**
with **#90** 19MB Winchester subsystem and one 80 track	**$8898.79**
with a 47MB Winchester subsystem and one 80 track	**$10,898.79**
with a 47MB plus a 6MB removable pack Winchester subsystem and one 80 track drive	**$12,398.79**

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

OS-9 GMX I / FLEX SYSTEMS #49

The **#49** systems include 64KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$3998.49**
with dual 80 track DSDD drives	**$4198.49**
with **#88** dual 8" DSDD drive system	**$5698.49**
with **#90** 19MB Winchester subsystem and one 80 track	**$6898.49**

OS-9 GMX II / FLEX SYSTEMS #39

The **#39** systems include 128KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$4498.39**
with dual 80 track DSDD drives	**$4698.39**
with **#88** dual 8" DSDD drive system	**$6198.39**
with **#90** 19MB Winchester subsystem and one 80 track	**$7398.39**

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Contact GIMIX for price and availability of UniFLEX and UniFLEX GMXIII Systems.

NOTE on all drive systems: Dual 40 track drives have about 700KB of formatted capacity; dual 80's about 1,400KB; dual 8" about 2,000KB. The formatted capacity of hard disks is about 80% of the total capacity.

Want to expand your system to a megabyte of Static RAM and 15 users?

Simply add additional memory and I/O boards. Your **GIMIX** system can grow with your needs. Contact us for a complete list of available boards and options.

#72 256KB CMOS STATIC RAM board with battery back up	**$1898.72**
#64 64KB CMOS STATIC RAM board with battery back up	**$528.64**
#67 64KB STATIC RAM board	**$478.67**
#11 3 port intelligent serial I/O board	**$498.11**
#43 2 port serial I/O board	**$128.43**
#42 2 port parallel I/O board	**$88.42**
#95 cable sets (1 needed per port), specify board	**$24.95**

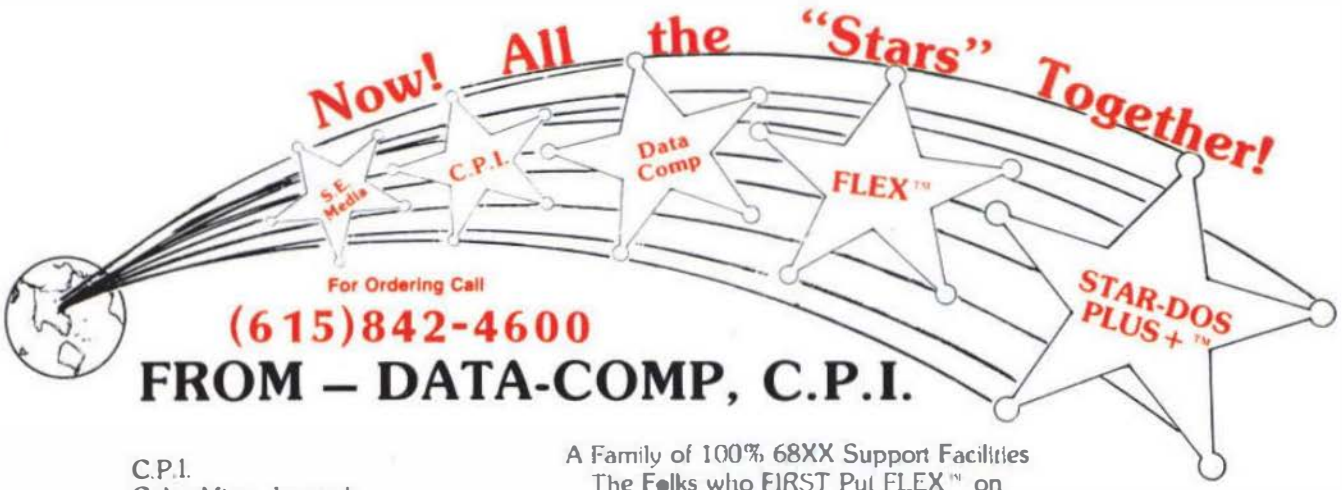
NOW SHIPPING !

UniFLEX

GMX III Systems

GIMIX INC.
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

©1984 GIMIX, INC. 4-84



For Ordering Call

(615)842-4600

FROM – DATA-COMP, C.P.I.

C.P.I.
Color Micro Journal
'68' Micro Journal
Data-Comp
S.E. Media

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo
Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00
Only $79.95

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.50

FLEX-CoCo Jr.
without TSC
Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ Many Many More!!!
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

CoCo Disk Drive Systems

NEW LOWER PRICES ON PAK #5, AND PRINTERS

THESE PACKAGES INCLUDE DRIVE, *CONTROLLER, POWER SUPPLY & CABINET, CABLE, AND MANUAL.

* SPECIFY WHAT CONTROLLER YOU WANT J&M, OR RADIO SHACK.

PAK #1 - 1 SINGLE SIDED, DOUBLE DENSITY SYS.	$349.95
PAK #2 - 2 SINGLE SIDED, DOUBLE DENSITY SYS.	$639.95
PAK #3 - 1 DOUBLE SIDED, DOUBLE DENSITY SYS.	$439.95
PAK #4 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS.	$699.95
PAK #5 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS. THINLINE DRIVES, HALF SIZE	$499.95

Controllers

J&M DISK CONTROLLER W/ JDOS OR RADIO SHACK DISK BASIC, SPECIFY WHAT DISK BASIC.	$134.95
RADIO SHACK DISK CONTROLLER 1.1	$134.95

Misc.

64K UPGRADE W/MOD. INSTRUCTIONS, C,D,E,F, AND COCO 2	$ 44.95
HJL KEYBOARDS	$ 74.95
MICRO TECH LOWER CASE ROM ADAPTER	$ 74.95
RADIO SHACK BASIC 1.2	$ 24.95
RADIO SHACK DISK BASIC 1.1	$ 24.95
DISK DRIVE CABINET & POWER SUPPLY	$ 49.95
SINGLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$199.95
DOUBLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$249.95

Printers

EPSON RX-80	$269.00
EPSON RX-80FT	$369.00
EPSON MX-100	$499.00
EPSON FX-100	$799.00
EPSON FX-80	$549.00
EPSON MX-70	$200.00

Disk Drive Cables

CABLE FOR ONE DRIVE	$ 19.95
CABLE FOR TWO DRIVES	$ 24.95

DATA-COMP
5908 Cassandra Smith Rd.
Hixson, TN 37343

VISA

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

(615)842-4600
For Ordering
TELEX 558 414 PVT BTH

68 Micro Journal
ISSN 0194-5025
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Hixson, TN
And Additional Mailing Offices

KINGSTON SPRINGS TN 37082
P. O. BOX 87
MICKEY FERGUSON

Advanced typing and text editing capability.

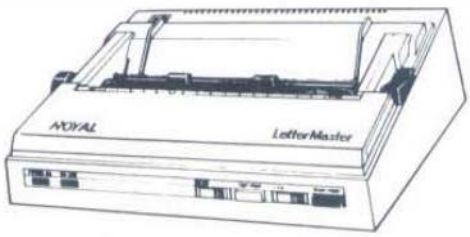
Dual-purpose asset for home and office.

ROYAL Beta 9000D Electronic Memory Typewriter with Display

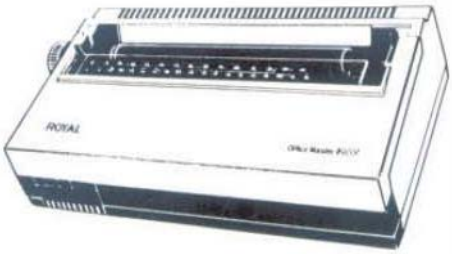
The perfect combination . . . advanced electronic typing and text editing capability. The ROYAL Beta 9000D features an easily accessed 2500 character phrase memory that lets you recall names, addresses and commonly used phrases at the touch of a key, a user-friendly 20-character display for ease of operation, 500 character lift-off correction memory, triple pitch, and much, much more. The Beta 9000D is also computer interfaceable via ROYAL's optional IF600 Interface Box with 4K memory. Use it as a sophisticated memory typewriter or as a letter quality computer printer. Either way, the ROYAL Beta 9000D delivers professional performance. See the Beta 9000D at:

ROYAL Beta 8200C Professional Portable Electronic Typewriter— Built-in Centronics Interface

The ROYAL Beta 8200C offers advanced electronic typing performance . . . with 2-line lift-off Correction Memory, Triple Pitch, 111-Character Keyboard with International Language, Math, Legal and Business Symbols, Automatic Indent, Center, Return, Decimal Tab and much, much more. The ROYAL Beta 8200C also features a built-in Centronics Parallel computer interface with 2K memory. Use it as a typewriter or as a letter quality computer printer. Either way, you get advanced performance and ROYAL value. See the Beta 8200C in action at:



Letter Quality 9 CPS
Dual Pitch Daisy Wheel



Letter Quality 20 CPS
Dual Pitch Daisy Wheel

Beta 9000 D	$ 599.95
Beta 8200 C	$ 499.95
Lettermaster	$ 239.95
Officemaster 2000	$ 499.95

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

VISA

SHIPPING
USA ADD 2%
FOREIGN ADD 5%

(615)842-4600
TELEX 558 414 PVT STN